GENNAIO 2018

RetroMagazine

Anno 2 - Numero 3

Editoriale: connubio retro/moderno
ARTICOLI
- Tra il DIR e il fare...
- Installare Windows 3.1 su VirtualBox
- Breve storia del TI-99 - parte 2
- Vite infinite con il C64
- RetroMath: i Frattali
- CBM prg Studio - C64 CrossDev Tool
GIOCHI
- Microsoft Decathlon (IBM PC)
- Actraiser (Super Famicom)
- Lotus Esprit Turbo Challenge (Amiga)
- Glorious Force (MSX)
- Dracula's Castle (C64)
INTERVISTE
- Intervista a Marco Pistorio
Sui prossimi numeri...

IN EVIDENZA IN QUESTO NUMERO

Editoriale: connubio retro/moderno

di Francesco Fiorentini

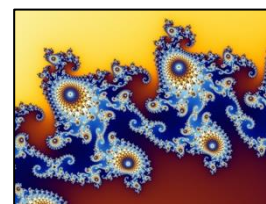
Bentornati cari lettori! Mi auguro che abbiate trascorso un Buon Natale, magari con la vostra famiglia come impone la tradizione e che i festeggiamenti dell'ultimo dell'anno siano stati all'altezza delle aspettative.

Un nuovo anno e' appena iniziato e noi di RetroMagazine ci apprestiamo a cominciare questo 2018 carichi di buoni propositi e di ottime intenzioni.

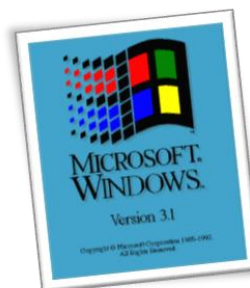
Prima pero' di vedere cosa ci riserva il nuovo numero, permettetemi di fare un veloce bilancio dei primi numeri della nostra rivista. Purtroppo non posso riportare i numeri del numero 1, ma il secondo numero ha fatto registrare ben piu' di 770 download. Un numero impressionante tenuto conto che la rivista si rivolge ad un pubblico di nicchia, tratta un argomento non particolarmente popolare al grande pubblico e viene distribuita solo tramite i canali Facebook od il passaparola. Colgo quindi l'occasione per ringraziare a nome dell'intera redazione ognuno di Voi. Noi ci mettiamo la passione e tutto l'entusiasmo possibile, ma siete Voi ed i Vostri feedback che ci spingono ogni mese a preparare qualcosa di unico nel suo genere ed a pensare che il prossimo numero sara' di nuovo un successo. Grazie!!!

Ma veniamo finalmente al numero 3.

Come avrete avuto modo di notare dal sommario, gli articoli pubblicati ed i giochi recensiti in questo numero non riguardano una macchina in particolare (anche se il C64 la fa sempre da padrone), ma cercano di spaziare a 360 gradi nel mondo del retrocomputing. Questo ci riempie di orgoglio, in quanto non vogliamo diventare una rivista mono sistema e cercando di abbracciare piu' sistemi possibile riuscire nel nostro intento di coinvolgere sempre piu' persone ad appassionarsi non solo al retrocomputing ma alla programmazione ed alla tecnologia in generale. Il codice dell'articolo sui frattali di Giuseppe e Marco e' facilmente trasportabile in un linguaggio moderno anche se pensato e scritto per il C64, mentre l'installazione di Windows 3.1 su una macchina virtuale e' possibile grazie alla potenza di calcolo delle macchine moderne. Per non parlare dell'articolo sul CBM Prg Studio di Giorgio, un esempio lampante di come si sposino retro e strumenti moderni.

Se non si fosse capito, amo questo connubio retro/moderno che ci permette di conciliare la nostra passione con l'avanzamento della tecnologia!

RetroMath: i Frattali

Insieme a **Giuseppe e Marco** entriamo nell'affascinante mondo dei frattali scoprendo cosa si nasconde dietro le belle immagini che costruiamo al computer e quali strumenti matematici ci servono per studiare queste nuove strutture.

Articolo a pagina 19

Installare Win 3.1 su VirtualBox

Sietre curiosi di vedere uno dei primi sistemi operativi creati da zio Bill ma non avete a disposizione l'hardware adatto? Scopriamo con Francesco come installare un sistema operativo vecchio di 25 anni su di una macchina virtuale.

Articolo a pagina 2

Tra il DIR e il fare...

di Francesco Fiorentini

Il comando DIR e' forse il primo comando DOS che ognuno di noi ha imparato a digitare una volta acceso il primo PC IBM. Ma quanti di noi hanno mai utilizzato a pieno tutte le opzioni del comando in questione?

Non voglio stare certo qui ad elencarle tutte ed a spiegare il funzionamento di ognuna di esse, il mio e' semplicemente un tentativo per farci riflettere sulle cose che spesso diamo per scontate. Il comando in questione e' banale nel suo scopo, ma nasconde dentro di se' tante peculiarita' che forse non tutti hanno avuto modo di testare. Quante volte siamo ricorsi a tool terzi parti quando invece con un po' di inventiva avremmo potuto risolvere con un semplice comando DOS e l'applicazione di pipe o ri-dirigerendo l'output su un file? Da questo punto di vista gli amministratori Unix e Linux hanno tanto da insegnare ai sistemisti DOS.

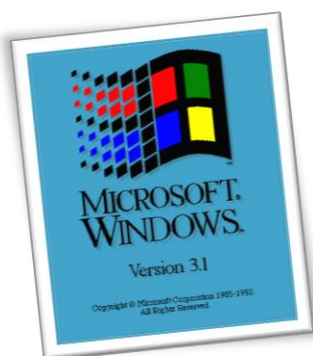
Ma perche' hai tirato in ballo proprio il comando DIR? E' un comando veramente semplice... Si', lo e', ma ha una lunga storia alle spalle e si puo' usare anche sul **C128**.

Curiosi? Continuate a seguirci!

```
  [/O[[:]sortorder]] [/P] [/Q] [/R] [/S] [/T[[:]timefield]] [/W] [/X] [/4]

  [drive:][path][filename]
              Specifies drive, directory, and/or files to list.

  /A          Displays files with specified attributes.
  attributes   D  Directories                R  Read-only files
               H  Hidden files               A  Files ready for archiving
               S  System files               I  Not content indexed files
               L  Reparse Points             -  Prefix meaning not
  /B          Uses bare format (no heading information or summary).
  /C          Display the thousand separator in file sizes.  This is the
              default.  Use /-C to disable display of separator.
  /D          Same as wide but files are list sorted by column.
  /L          Uses lowercase.
  /N          New long list format where filenames are on the far right.
  /O          List by files in sorted order.
  sortorder    N  By name (alphabetic)       S  By size (smallest first)
               E  By extension (alphabetic)  D  By date/time (oldest first)
               G  Group directories first    -  Prefix to reverse order
  /P          Pauses after each screenful of information.
  /Q          Display the owner of the file.
  /R          Display alternate data streams of the file.
  /S          Displays files in specified directory and all subdirectories.
  /T          Controls which time field displayed or used for sorting
  timefield   C  Creation
              A  Last Access
              W  Last Written
  /W          Uses wide list format.
  /X          This displays the short names generated for non-8dot3 file
              names.  The format is that of /N with the short name inserted
              before the long name. If no short name is present, blanks are
              displayed in its place.
  /4          Displays four-digit years

witches may be preset in the DIRCMD environment variable.  Override
reset switches by prefixing any switch with - (hyphen)--for example, /-W.
```

Avete mai posto attenzione alla miriade di opzioni presenti nell'help del comando DIR? Davvero interessante...

Installare Windows 3.1 su VirtualBox

di Francesco Fiorentini

Scopriamo come utilizzare VirtualBox e le macchine virtuali per installare un sistema operativo di 25 anni fa! Windows 3.1.

Installare Windows 3.1 su una macchina virtuale? A cosa può servire Windows 3.1 nel 2017? Beh, onestamente a poco, ma con questo articolo voglio introdurre alcuni concetti correlati con il retro-computing, ossia l'emulazione e le macchine virtuali. I puristi del retro-computing storceranno la bocca all'idea di emulare un macchina storica, preferendo di gran lunga l'esperienza di gioco/coding sulle macchine reali (ed io concordo), ma purtroppo per le ragioni più disparate, non tutti hanno a disposizione un parco macchine completo dove sperimentare i giochi e gli applicativi che vorrebbero testare. Inoltre, le nostre amate macchine hanno ormai decine di anni di vita sulle spalle e sebbene l'utilizzo gli sia salutare, in alcuni casi potremmo volerle risparmiare... A questo proposito ci vengono incontro gli emulatori e le macchine virtuali.

Ambedue i concetti sono ormai stranoti, ma per chi si avvicinasse al mondo dell'emulazione e della virtualizzazione per la prima volta è giusto fornirne una descrizione.

Emulazione

In informatica, la procedura con cui un elaboratore riproduce in un sistema di elaborazione il funzionamento di un altro elaboratore di analoghe o inferiori capacità tecniche. La definizione riportata dal dizionario è perfetta, ma potrebbe non essere così chiara per tutti. In pratica, il vostro computer, tramite un apposito software chiamato emulatore, riproduce il

funzionamento di un altro computer ed è quindi in grado di eseguire i comandi, le istruzioni ed il software del computer emulato. Senza andare troppo nel tecnicismo, immaginate l'emulatore come una specie di *macchina virtuale* all'interno del vostro computer in grado di emulare il funzionamento di un altro computer/console. Ritorneremo al concetto di emulatore nei prossimi numeri di RetroMagazine, promesso.

Macchina Virtuale.

In informatica il termine macchina virtuale (VM) indica un software che, attraverso un processo di virtualizzazione, crea un ambiente virtuale che emula tipicamente il comportamento di una macchina fisica (PC client o server) grazie all'assegnazione di risorse hardware (porzioni di disco rigido, RAM e risorse di processamento) ed in cui alcune applicazioni possono essere eseguite come se interagissero con tale macchina; infatti se dovesse andare fuori uso il sistema operativo che gira sulla macchina virtuale, il sistema di base non ne risentirebbe affatto. (Fonte *Wikipedia Italia*).

Figura 2

Come vedete i due concetti sono molto simili, quale è quindi la differenza tra un emulatore ed una macchina virtuale? Praticamente nell'emulazione le istruzioni assembly della macchina ospitata vengono tradotte in istruzioni comprensibili alla macchina ospitante, mentre nel concetto di virtualizzazione software (come quello implementato in Virtual Box che vedremo di seguito), è l'hypervisor (il cuore della macchina virtuale) a creare soltanto un ambiente isolato nella macchina ospitante dove far girare l'ospite. In questo caso generalmente le istruzioni delle due macchine non sono tradotte in quanto l'HW delle stesse è compatibile.

Dopo la breve ma doverosa introduzione, passiamo a spendere sue parole su Windows 3.1 e capire perchè l'ho scelto per un articolo dedicato alla virtualizzazione. Windows 3.1 viene rilasciato da Microsoft nel 1992 (in US) ed è il primo sistema operativo grafico pre-installato, insieme al DOS, sui nuovi PC. Da Windows 3.1 in poi comincerà a registrarsi la tendenza ad abbandonare l'interfaccia a caratteri tipica del DOS (giunto ormai alla versione 6) per utilizzare invece la più comoda, per l'utente medio, interfaccia grafica del nuovo ambiente della casa di Redmond. Nonostante abbia ancora la necessità di essere avviato da riga di comando del DOS e quindi non ancora un vero e proprio sistema operativo standalone, è il trampolino di lancio per Microsoft per un sistema operativo completamente a 32bit che prenderà il nome di Windows NT (New Technology). Con Windows 3.1 fanno inoltre la loro comparsa i Font True Type ed i primi rudimentali strumenti multimediali, rendendo alla portata degli utenti PC alcune funzionalità ancora ad appannaggio dei sistemi Apple e Commodore. Un software rudimentale quindi, ma che ha gettato le basi per la conquista della leadership del mercato PC da parte di Microsoft.

Ovvia Francesco, vuoi farci vedere qualcosa o dobbiamo ancora sorbirci una lezione di storia dell'informatica per tutto l'articolo? Tranquilli, la lezioncina è terminata ed è giunto il momento di fare qualcosa di pratico. Controlliamo di avere tutto l'occorrente a portata di mano e cominciamo ad installare Windows 3.1 su Virtual Box:

Figura 1

Virtual Box scaricabile da:

https://www.virtualbox.org/

MS DOS 6.0 scaricabile da:

https://winworldpc.com/product/ms-dos/6x

Windows 3.1 scaricabile da:

https://winworldpc.com/product/windows-3/31

Il motivo per cui ho scelto Virtual Box è piuttosto semplice, è versatile, professionale ed inoltre è distribuito come Open Source Software sotto la licenza GNU General Public License (GPL) version 2, che permette in pratica di utilizzare il software a fini privati e professionali gratuitamente. Sorvolerò sull'installazione di Virtual Box; è un software moderno dove si tratta in pratica di cliccare avanti, avanti, avanti o poco più, quindi non credo ci siano difficoltà da parte di nessuno a portare a termine l'installazione (se avete comunque problemi potete contattarmi tramite la mailbox redazionale).

Creiamo una macchina virtuale

Una volta avviato Virtual Box, dobbiamo creare una nuova macchina virtuale, icona '**Nuova**' e tenendo in mente i requisiti minimi di Windows 3.1 possiamo configurarla adeguatamente ad ospitare il sistema operativo. I requisiti minimi di Windows 3.1 sono i seguenti: MS-DOS 3.3 o successivo, CPU 286 o superiore, 3 MB RAM e 9 MB di spazio libero su disco.

Figura 3

Noterete subito che Virtual Box non appena date alla macchina il nome Windows 3.1 selezionerà automaticamente il sistema operativo ospite e suggerirà la quantità di memoria RAM da allocare in 32 Mbytes (adesso ci fa ridere, ma vi assicuro che nel 1992 era un sacco di memoria RAM :-)). Continuiamo pure così e selezioniamo 'Crea subito un nuovo disco fisso virtuale' (vedi Figura 1) e clicchiamo 'Crea'. A questo punto dobbiamo specificare la dimensione e la posizione dell'Hard Disk virtuale da dedicare al nostro sistema operativo. Scegliete pure un percorso consono sul vostro Hard Disk e lasciate invariate la dimensione di 1 GB ed il formato VDI così come suggerito da Virtual Box (vedi Figura 2).

Figura 4

A questo punto la nostra macchina virtuale è pronta per essere avviata e di conseguenza pronta a ricevere l'installazione del DOS 6.22 e di Windows 3.1.

In teoria ci sono molte impostazioni che possono essere configurate sulla macchina virtuale, tanti e tali ed inoltre dipendenti dall'hardware ospitante, che richiederebbero un apposito numero di RetroMagazine per essere approfondite. Inoltre esulano un po' dallo scopo della nostra rivista e quindi passiamo al punto successivo.

Installiamo il DOS 6.0

Come anticipato nel cappello introduttivo, il sistema operativo Windows 3.1 non è standalone, ma si basa sull'architettura del DOS, quindi prima di procedere ad installare Windows, dobbiamo creare lo strato sottostante installando una versione di DOS a partire dalla 3.3. Per comodità ho scelto la versione 6.0.

Avviando la macchina virtuale tramite il comando 'Avvia', vi dovreste trovare di fronte la schermata di Figura 3. In pratica la vostra macchina virtuale non ha trovato nessun sistema operativo e quindi non sapendo come comportarsi restituisce il messaggio di errore '**FATAL: No bootable medium found! System halted.**' Tramite il comando 'Dispositivi' -> 'Lettori Floppy' scegliete il primo disco contenuto nello zip del DOS 6.0 e riavviate la macchina... Decisamente meglio vero? La vostra macchina virtuale si è avviata tramite il primo disco dell'MS DOS ed è pronta per installarlo (vedi Figura 4). Scegliamo quindi di formattare l'HD virtuale configurando lo spazio non allocato e proseguiamo con l'installazione. Dopo aver provveduto a formattare il disco (non vi preoccupate è il disco virtuale da 1GB che abbiamo creato in figura 2) ed aver selezionato la nazione ed il layout della tastiera (Figure 5 e 6), confermiamo pure la

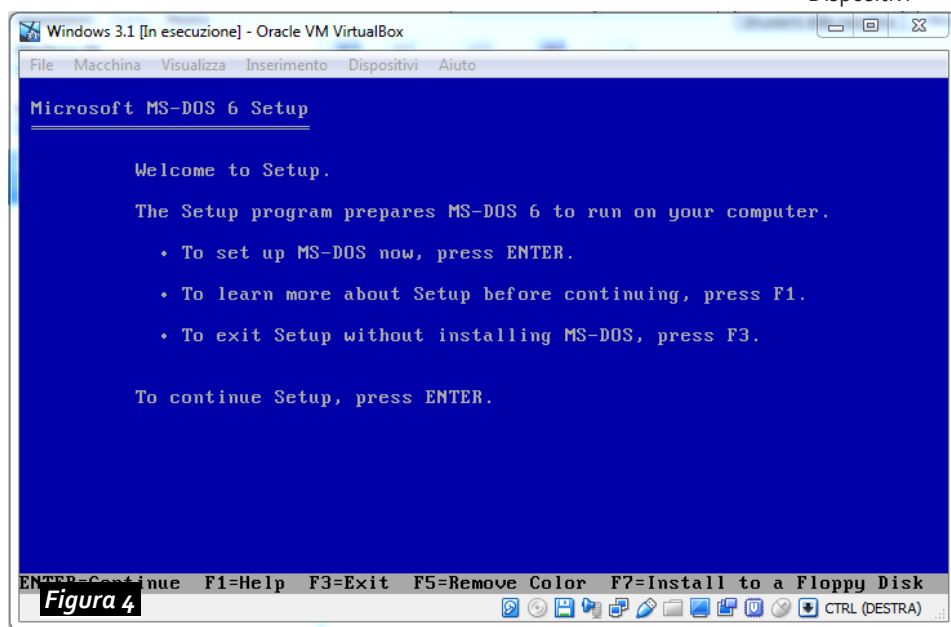



Figura 5

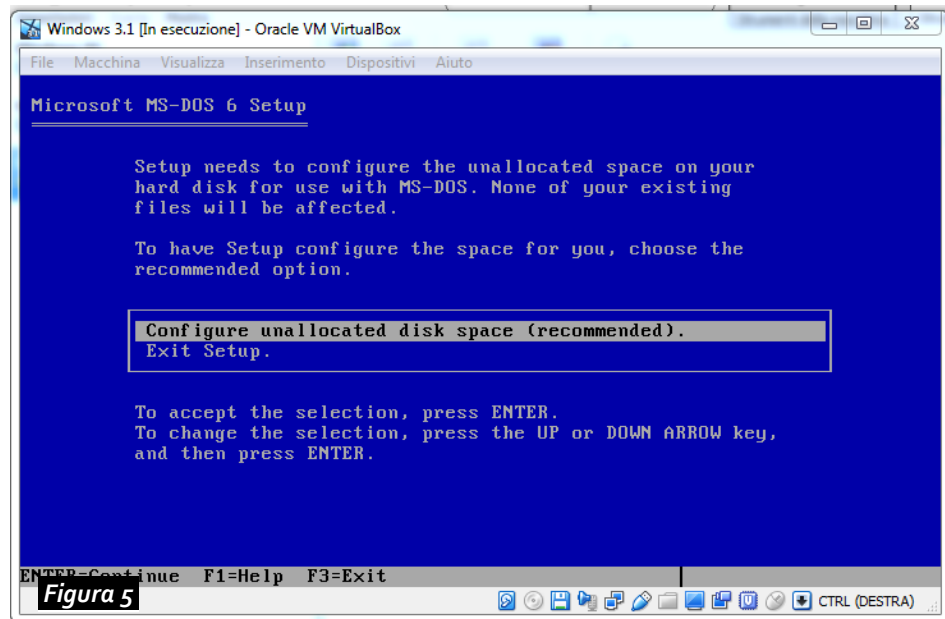
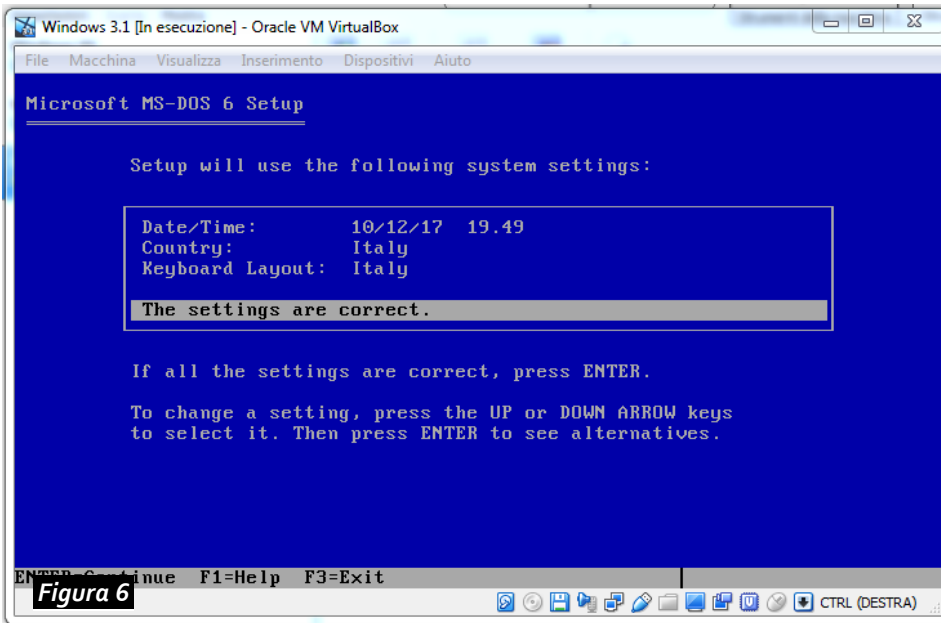
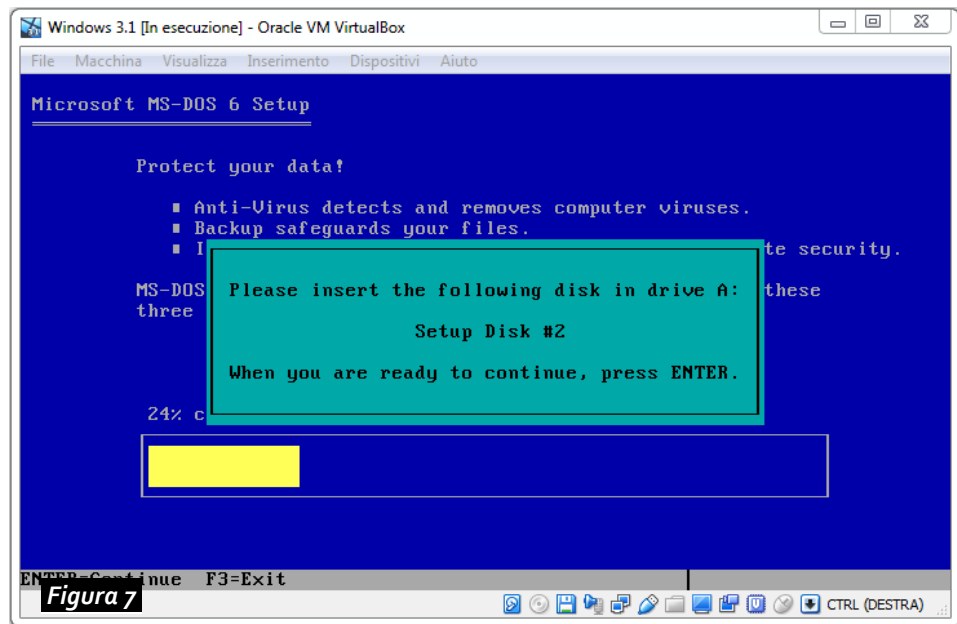



Figura 6

directory C:\DOS come destinazione e lasciamo il programma di installazione a fare il resto. Il programma vi richiederà un paio di volte di cambiare dischetto per inserire i dischi 2 e 3 (Figura 7), fatelo tranquillamente con il comando 'Dispositivi' -> 'Lettori Floppy' scegliendo il disco opportuno e dopo qualche secondo, dopo aver rimosso tutti i dischetti dal lettore floppy ed aver riavviato, vi ritroverete con il DOS 6.0 installato sulla vostra macchina virtuale (vedi Figura 8). Complimenti, il primo passo è fatto, la base per installare Windows 3.1 è pronta.

Installiamo Windows 3.1

Inseriamo il dischetto numero 1 di Windows 3.1, scegliendolo tra i vari dischi scompattati dal file zip scaricato da winworldpc.com e digitando il comando **A:** (seguito da invio) ci sposteremo all'interno dello stesso. A questo punto scriviamo il fatidico comando **SETUP** seguito da invio e diamo inizio al processo di installazione (Figura 9).

La schermata iniziale dell'installazione di Windows 3.1 è identica a quella del DOS, niente a che vedere con le installazioni grafiche a cui siamo abituati oggigiorno, premiamo invio per avviare il processo e poi di nuovo invio per scegliere l'express setup. Prepariamoci ad un lavoro manuale degno di un disc jokey, dovremmo cambiare disco diverse volte e ci sono ben 6 dischi da installare. :-) Dopo aver letto il secondo dischetto Windows vi chiederà di specificare il nome utente e la società di appartenenza, indicate pure quello che volete (Figura 10). Dopo aver esaurito tutti i dischi a disposzione, il setup vi richiederà di scegliere una stampante, premete pure **CANCEL**, e di specificare il nome del da abbinare al comando C:\DOS\EDIT.COM, confermate pure MS-DOS Editor e premete OK, saltate il tutorial di presentazione di Windows, rimuovete il dischetto dal lettore floppy e premete REBOOT (Figura 11).

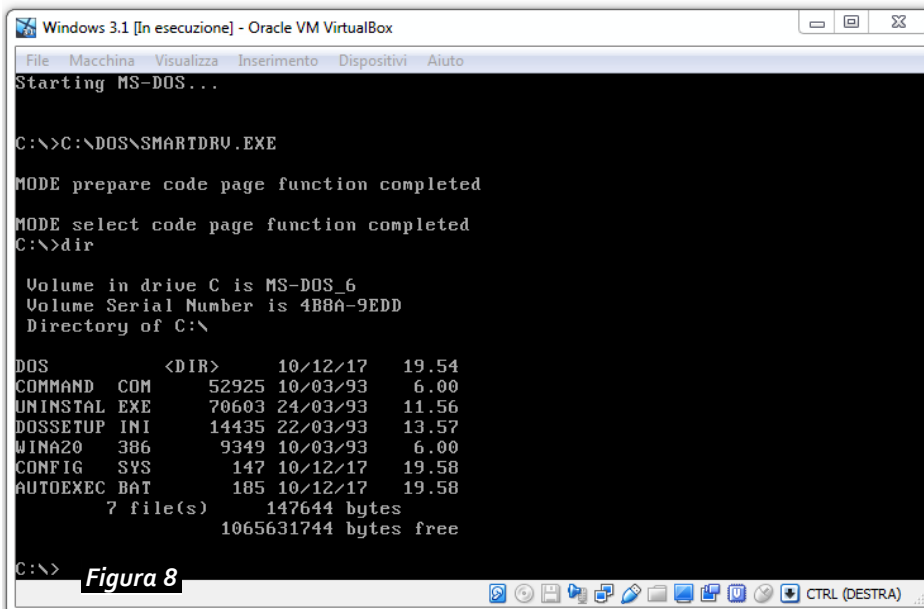



Figura 7

Windows 3.1 è a questo punto installato sull'Hard Disk della vostra macchina virtuale... Ehi, ma che succede, ho riavviato la macchina e mi ritrovo ancora al prompt del DOS, perchè non è partito Windows? Perchè Windows 3.1, come detto sopra, non è un SO standalone e deve essere lanciato dal prompt del DOS tramite il comando **WIN**. Scrivete quindi WIN seguito da invio ed in una manciata di secondi dovreste torvarvi alla schermata di Figura 12.

Windows 3.1 [In esecuzione] - Oracle VM VirtualBox

File Macchina Visualizza Inserimento Dispositivi Aiuto

```
Starting MS-DOS...

C:\>C:\DOS\SMARTDRV.EXE

MODE prepare code page function completed

MODE select code page function completed
C:\>dir

 Volume in drive C is MS-DOS_6
 Volume Serial Number is 4B8A-9EDD
 Directory of C:\

DOS          <DIR>     10/12/17   19.54
COMMAND  COM     52925 10/03/93    6.00
UNINSTAL EXE     70603 24/03/93   11.56
DOSSETUP INI     14435 22/03/93   13.57
WINA20   386      9349 10/03/93    6.00
CONFIG   SYS       147 10/12/17   19.58
AUTOEXEC BAT       185 10/12/17   19.58
        7 file(s)     147644 bytes
                  1065631744 bytes free

C:\>
```

Figura 8

CTRL (DESTRA)

Precisazione

Trattanadosi di una guida destinata ad una utenza di livello medio, alcuni passaggi potrebbero essere stati omessi o descritti molto velocemente. Se avete domande in merito all'articolo vi invito a scrivere alla mailbox redazionale: RetroMagazine.redazione@gmail.com

Bene, eccoci arrivati alla fine di questo primo tutorial sul mondo delle macchine virtuali e su Windows 3.1. Potete cominciare a divertirvi con il vostro SO virtuale con la consapevolezza che qualsiasi cosa facciate potrete sempre ripristinare uno stato precedente, oppure potete aspettare i prossimi numeri di RetroMagazine per scoprire cosa possiamo ancora fare con Windows 3.1.

Curiosita' - provate ad aprire il File Manager e vi renderete subito conto di quanta strada Microsoft e Windows hanno fatto in 25 anni. E' vero, Windows 3.1 e' un sistema operativo grafico, ma alcuni programmi sembrano ancora delle mere trasposizioni dei loro cloni a caratteri. Il Fle manager non vi ricorda forse il famoso Norton Commander? (Figura 13).

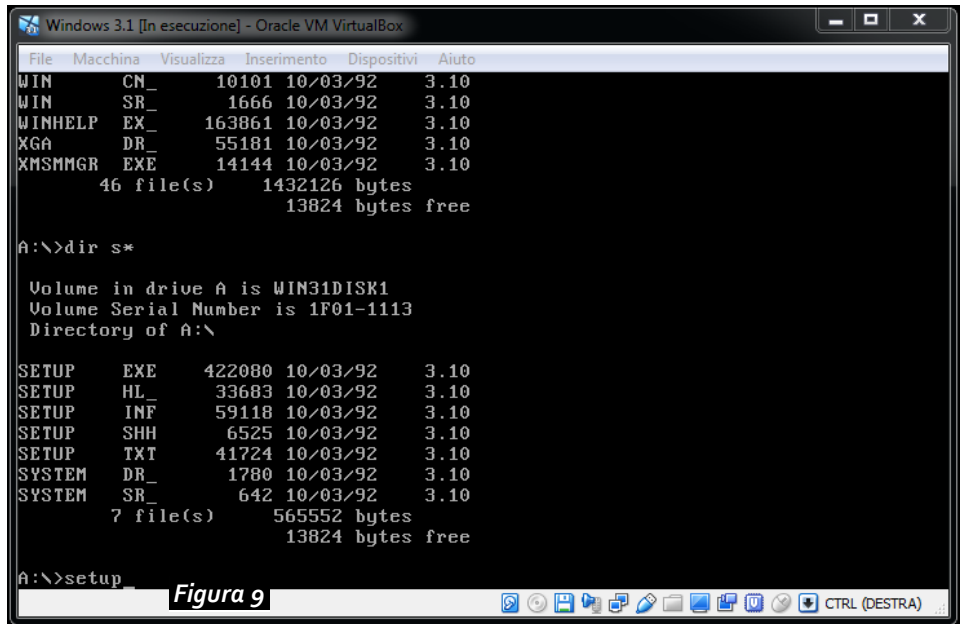



Figura 9

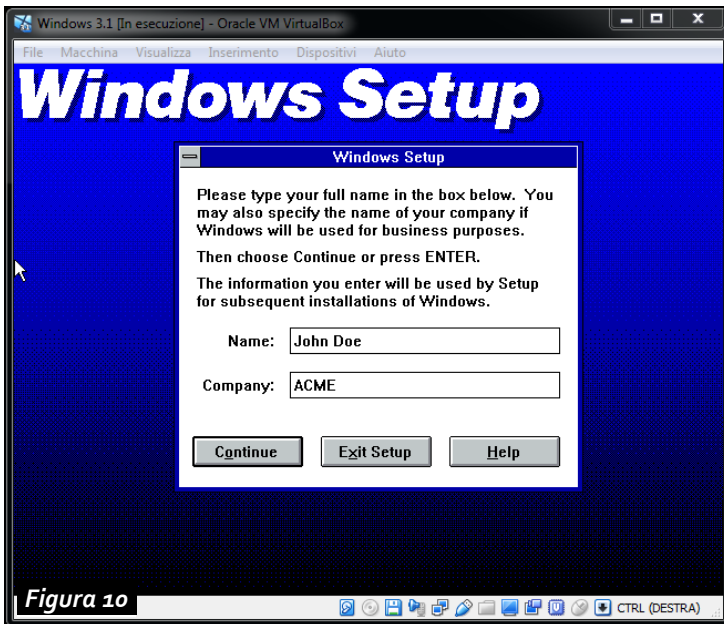



Figura 10

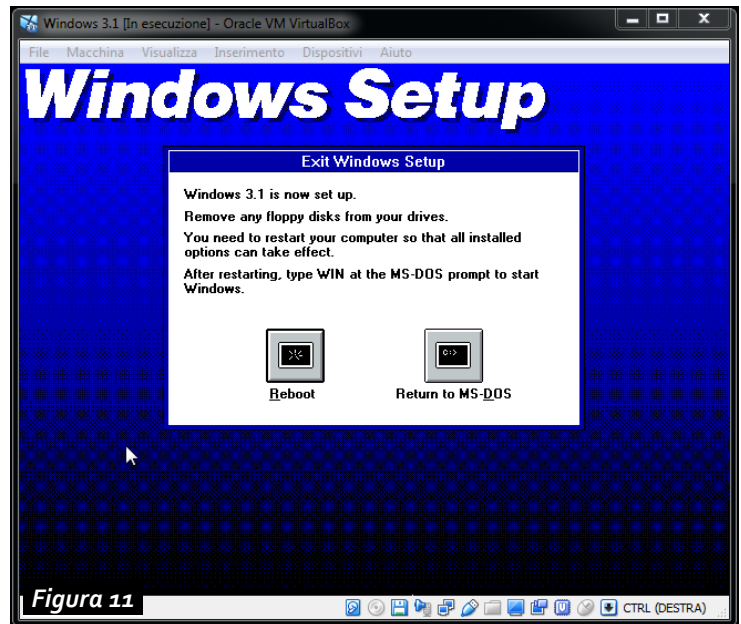



Figura 11

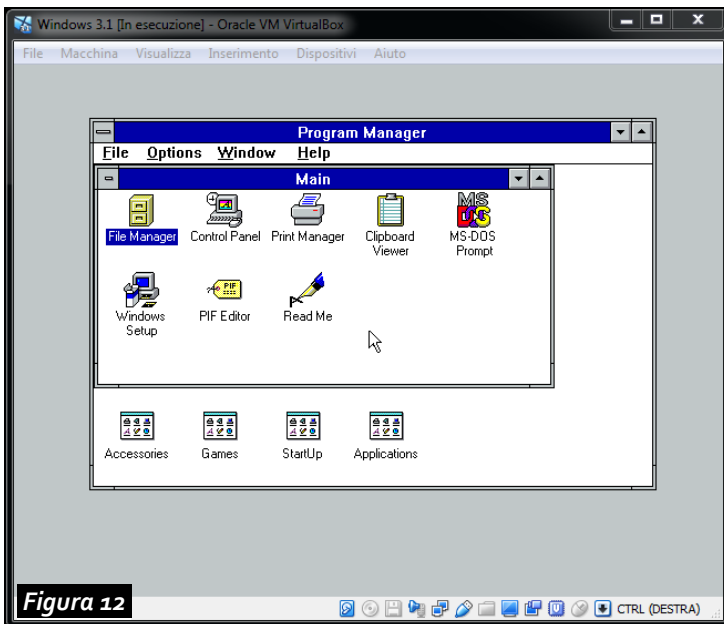



Figura 12

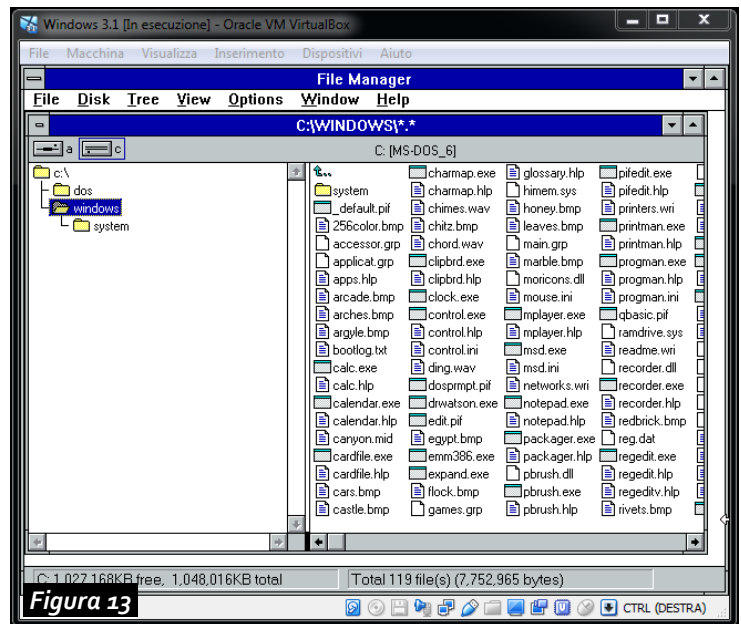
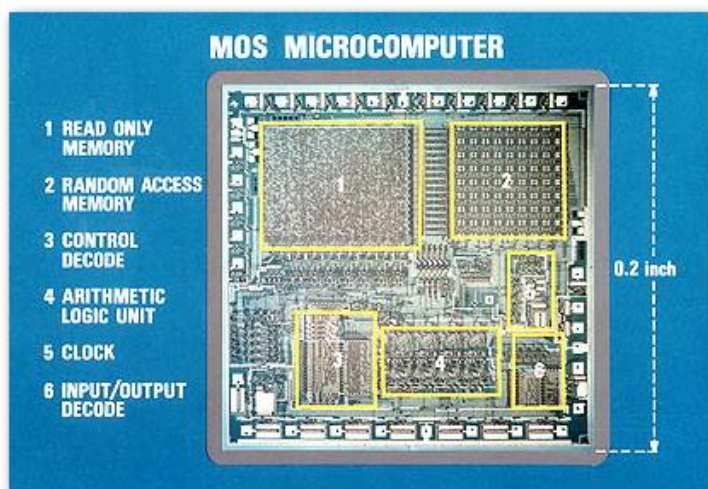



Figura 13

Breve storia del TI-99 – parte 2

...continua dal numero 2 - dal 1974 al 1983

di Ermanno betori

TI ha mostrato il TMS 1000, il primo microcomputer su un chip, nel primo trimestre del 1975 alla riunione degli azionisti.

1974 - Il Microcomputer TMS1000 a chip singolo vede la luce!

Nel 1974, TI introdusse il TMS 1000, un microcomputer a chip singolo che diede alla Texas Instruments la leadership sul mercato per applicazioni a consumo a 4 bit, automobilistico e industriale.

L'affidabilità era stata migliorata grazie alla costruzione di un chip che è più economico fino al 75 percento dei precedenti processori, il che contribuì a ridurre i costi complessivi del sistema. Il supporto totale della TI si concentrò ulteriormente verso il settore sviluppo, inclusi corsi di formazione per i clienti, sviluppo di software e staff di ingegneri delle applicazioni e specialisti in applicazioni sul campo. Il microcomputer fù utilizzato in applicazioni quali sistemi di comunicazione, giochi elettronici, distributori automatici e registratori di cassa, terminali e periferiche, controlli industriali, pompe di gas e verificatori di carte di credito. Per capire quale importanza aveva la TI nella opinione pubblica della società civile americana degli anni 70, la possiamo paragonare ad un simbolo come la statua della libertà, il dollaro, l' hotdog! La TI era l'america! Come fiducia rappresentava quasi un pilastro istituzionale.

1975 - Viene rilasciato il TMS9900, chip a microprocessore a 16 bit

Nel 1975, la TI era pronta a divulgare la sua famiglia di microprocessori a 16 bit e minicomputers compatibili software 9900/990, il cuore della sua strategia di integrazione verticale. Questa strategia permise la crescita di una task force tecnica orientata verso i computer di nuova generazione. Il TMS9900 è stato il primo microprocessore a 16 bit dell'industria. La famiglia 9900/990 venne progettata con un'architettura innovativa memoria/memoria.

Questi prodotti erano al centro di una strategia innovativa che sperava di creare un business completamente nuovo. Nel 1976, la società rese ufficiale le proprie ambizioni: Il computer distribuito sarebbe stato uno dei tre capisaldi della TI. Nel 1976, il TI 990 diventò il primo in una crescente famiglia di minicomputers basata sulla compatibilità del software con il microprocessore a 16 bit di TI.

L'evoluzione della famiglia 9900 ha continuato con i microprocessori 9995 e 99000 ma si è conclusa nel 1982, quando emerse che il futuro successo del microprocessore della TI richiedeva una differenziazione da quello che facevano le altre società. Una squadra di nuova formazione abbracciò un concetto chiamato architettura funzione-a-funzione che enfatizzava i microprocessori su misura per particolari funzioni del sistema. L'elaborazione del segnale digitale, le reti locali, l'aritmetica a virgola mobile e la grafica sono stati selezionati come aree di messa a fuoco.

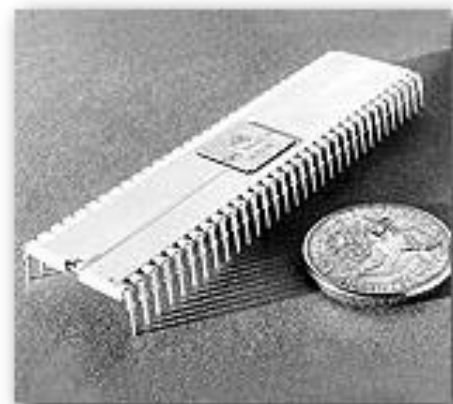
1978 - Il sintetizzatore vocale a chip singolo si esibisce nell'apprendimento TI Speak & Spell™

Un ulteriore sviluppo della ricerca della TI fu nell'ambito della voce digitale, il prodotto didattico Speak & Spell ™ fu progettato per aiutare i bambini dall'età di sette anni e più anziani a imparare,cantare e pronunciare più di 200 parole comunemente sbagliate.

Nel 1976 con uno studio di fattibilità di tre mesi e con un budget di $ 25.000, quattro ingegneri Tiers iniziarono il progetto. Paul Breedlove, Richard Wiggins, Larry Brantingham e Gene Frantz.

Il concetto di Speak & Spell crebbe dal brainstorming di Breedlove per i prodotti che dovevano dimostrare le capacità della memoria a bolle (un progetto di ricerca TI), e conclusero che i dati necessari per ricostruire foneticamente le frasi di un discorso avrebbero preso molta memoria e sarebbero stati una buona applicazione.

L'aiuto all'apprendimento parlando usava un concetto completamente nuovo nel riconoscimento vocale. A differenza dei registratori a nastro e delle registrazioni in stringhe utilizzate in molti giocattoli "parlanti", i circuiti di Solid State Speech di TI non avevano parti in movimento. Quando uno parlava, l'elaborazione avveniva attraverso un circuito integrato che ricostruiva la voce umana per poi pronunciarla elettronicamente. Segna la prima volta che il timbro vocale umano veniva duplicato elettronicamente su un unico chip di silicio. Il prodotto di consumo fu introdotto negli spettacoli Summer Electronics Consumer nel giugno 1978.

Il successo di Speak & Spell estese la spinta della TI verso i prodotti didattici Speak & Math ™, Speak & Read ™, Speak & Music ™ e un'intera collezione di giocattoli per

Nel 1975, TI ha introdotto il primo microprocessore dell'industria che utilizzante un linguaggio software compatibile con una famiglia di minicomputer, il 9900 a 16 bit, qui illustrato accanto a un pezzo di 25 centesimi per il confronto delle dimensioni.

bambini. Speak & Spell è andato in giro per il mondo - in diverse lingue.

Anche se è stato introdotto quasi 40 anni fa, i principi di apprendimento fondamentali e i concetti di progettazione di Speak & Spell restano lo standard per i giocattoli educativi. Le applicazioni di sintesi vocale e di riconoscimento vocale sono pervasive oggi - che vanno dalle applicazioni telefoniche per il controllo degli orari delle compagnie aeree, ai sistemi di navigazione vocali in automobili, ai computer per i ciechi e alle applicazioni di sicurezza.

1979 – Nasce il TI 99/4 Home Computer.

Nei primi anni '70, il presidente Pat Haggerty e il suo team di gestione avevano molte discussioni strategiche sul potenziale di prodotti elettronici a basso costo creati con i transistor o circuiti integrati che rendessero possibile un computer per la casa con implicazioni per usi commerciali, educativi e ricreativi. Queste discussioni portarono ad un progetto di computer di casa. Dopo il successo commerciale di calcolatori TI, il computer domestico sembrava il passo successivo logico. Il computer domestico diventò la scommessa più grande della TI nella ricerca dell'integrazione verticale, ma l'impresa si è rivelò un buco nero. L'enorme impatto negativo sulla salute finanziaria della TI avrebbe potuto affondare l'azienda, che si salvò grazie al forte bilancio, alla posizione leader nella vendita di componenti tecnologici ed alla sua forte area sviluppo che è sempre stato vanto di questa industria.

I visitatori del Consumer Electronics Show nel giugno 1979 furono in grado di vedere il computer portatile 99/4 della TI in funzione, confermando le voci che avevano circolato per mesi. Il suo processore centrale era il microprocessore TMS9900 a 16 bit di TI, progettato per essere una spanna sopra su altri computer a 8 bit presenti sul mercato. Dotato di un monitor a colori da 13 pollici, progettato per utilizzare moduli plug-in plastici di memoria di sola lettura, che contenevano programmi quali giochi, finanza personale e software didattico (quella che poi verrà definita comunemente cartuccia), utilizzo come memoria di massa a basso costo il tape, venne accolto con una reazione in gran parte favorevole, anche se alcuni furono sorpresi dal prezzo elevato di 1.150 dollari.

Speak & Spell, conosciuto in Italia come Sapientino...

Le risposte dal mercato indicarono la necessità di ulteriori titoli software, una tastiera migliore, più periferiche e un prezzo più basso. Il computer domestico riprogettato, il denominato TI99/4A, venne introdotto nel giugno 1981, a un prezzo di 525 dollari.

Un importante programma di marketing venne creato per incrementare le vendite. Il canale primario di distribuzione fu spostato dai negozi di computer a punti vendita ad alti volumi. Un team di oltre 2.000 insegnanti di scuole venne reclutato e addestrato per dimostrare il prodotto nei negozi. Una campagna di annunci creativi venne sviluppata usando Bill Cosby come portavoce. Il Gruppo Internazionale di utenti 99 / 4A diventò un'importante fonte di aggiunte e migliorie alla libreria di software che era in rapida crescita. Mentre la TI acquisiva quota di mercato e volume, il prezzo del 99/4A al dettaglio scese a 299 dollari per soddisfare le crescenti pressioni dei concorrenti.

1979 nasce il TI 99/4 Home Computer

Nell'aprile del 1982, la TI riferì di avere una posizione di rilievo nel mercato domestico emergente, con più di 5.000 punti vendita che si occupavano o si impegnavano a supportare il 99/4A. Sebbene il mercato fosse stato lento a svilupparsi, aveva cominciato ad espandersi rapidamente nel 1982 e la produzione era aumentata.

Nell'agosto 1982, le vendite di 99/4A erano in rapida crescita, ma il successo del computer domestico stava causando una forte concorrenza sui prezzi (famosa la guerra Atari vs Commodore). TI decise di offrire un rimborso di $ 100.

All'inizio del 1983, la TI aveva prodotto e venduto il milionesimo computer TI99/4A. La rete di distribuzione si era ampliata a 20.000 punti vendita in tutto il mondo. Erano disponibili più di 2.000 pacchetti software, sviluppati in gran parte da autori di terze parti, e vennero stipulati accordi con diverse software house specializzate nel campo educativo che ludico.

Poi, il disastro. Venne scoperto che in determinate circostanze, una persona avrebbe potuto ricevere uno shock elettrico dall'alimentatore del computer, anche se nessun utente aveva ancora provato un simile problema. La TI per non perdere reputazione (i suoi dipendenti erano orgogliosi di far parte di quella azienda a partire dai dirigenti fino all'ultimo impiegato), si armarono con i fatti, riferirono alle autorità di regolamentazione della sicurezza e ha decisero di interrompere temporaneamente il programma di vendite 99/4A finché i nuovi alimentatori non sarebbero stati prodotti e spediti a TI. Tutte le scorte al dettaglio e le richieste dei clienti di 99/4A vennero rielaborate e restituite. TI accumulò 50 milioni di dollari di passivo come costo stimato per risolvere il problema. Ancora più dannoso fù la mancanza di due mesi della TI dal mercato al dettaglio, permettendo ad altre case di emergere.. Commodore su tutte la quale aveva una vendetta personale da presentare alla Texas Instruments.

Venne formato un team di salvataggio di manager esperti del TI e inviato a Lubbock, Texas, sede e sito produttivo per il computer di casa. Questa squadra insieme al team del computer TI994A già presenti in Lubbock fecero uno sforzo intensivo per risolvere il problema del trasformatore, ripristinare lo slancio delle vendite al dettaglio, regolare la produzione e le scorte e ridurre i costi, ma affrontarono una situazione difficile.

1983 – TI Annuncia l'uscita dal mercato del computer casalingo

Il 28 ottobre 1983, la TI annunciò a sorpresa di molti di ritirarsi dal mercato dei computer casalinghi. Per circa un anno la TI aveva creato una cortina fumogena per impedire speculazioni finanziarie (oltre che per orgoglio personale), creando nuovi modelli di computer che non entrarono mai in vendita, nuovi programmi ecc.. allo scopo di non rivelare il suo vero stato di salute finanziario. Ma quando il fumo venne eliminato e i rendiconti finanziari vennero alla luce la TI dovette riferire agli azionisti che le perdite di segmento totale del computer casalingo erano di circa 680 milioni di dollari, incluse perdite operative, svalutazioni,

svalutazioni e altri accantonamenti associati alla chiusura.

Il ritiro dal mercato colpì le finanze di TI, ma i danni alla sua psiche e alla sua reputazione erano altrettanto dolorosi. Vi furono scene di panico di massa.. le persone pensavano che il computer smettesse di funzionare all'istante ecc... la descrizione di quei momenti viene descritta molto bene nel libro TI Orphan Chronical era la fine di un mondo...

Molti si preoccuparono di ciò che sarebbe accaduto con l'uscita di mamma TI dal mercato, pensando che avrebbe abbandonato al loro destino i milioni di utenti del loro computer.

Invece la Texas Instruments fece quello di fatto verrà giudicato dai posteri un testamento al carattere della società: La TI decise di continuare a sostenere gli utenti 99 / 4A durante il ritiro, onorare le garanzie e aiutare i concessionari con le loro scorte. Era un grosso prezzo da pagare, ma la TI pensava che fosse la cosa giusta da fare. Il calcolatore TI99 era nato per essere un mezzo educativo e le attività di apprendimento aiutano a crescere.. non era corretto abbandonare gli utenti..

Anche con l'uscita dal mercato nel 1983 da parte della TI, l'home computer TI99/4A è stato quello più diffuso, nel corso del prossimo anno o giù di lì, la maggior parte dei principali produttori di home computer lascieranno tranquillamente il mercato dopo aver sperimentato le proprie delusioni. Molti anni dopo, una serie di prodotti digitali piuttosto diversi sono emersi su misura per servire specifici sottosegmenti del mercato interno, tra cui giochi, istruzione, comunicazione e applicazioni per piccole imprese / home office. La TI continua a fornire soluzioni elettroniche/informatiche ai propri clienti in molte di queste applicazioni, con una preziosa prospettiva guadagnata dalle dolorose lezioni dell'esperienza di mercato del home computer. Questa nota informativa è solo la prima parte della trilogia presentiva del TI994A.

Nei prossimi numeri trattero' tutto l'hardware ed il software della macchina dal 1978 fino ai giorni nostri.

TEXAS INSTRUMENTS INCORPORATED

Stato	**Stati Uniti (USA)**
Forma societaria	**Societa' per azioni**
Borse valori	**NASDAQ: TXN**
ISIN	**US8825081040**
Fondata	**1930 (come GSI)** **1951 (Texas Instruments)**
Fondata da	**Cecil Howard Green**
Sede	**Dallas (Texas)**
Settore	**Elettronica**
Fatturato	**13 miliardi di $ (2016)**
Dipendenti	**circa 30.000 (2016)**
Sito Web	**www.ti.com**

FONTE WIKIPEDIA PAGINE ITALIANA ED INGLESE:
HTTPS://IT.WIKIPEDIA.ORG/WIKI/TEXAS_INSTRUMENTS_INCORPORATED
HTTPS://EN.WIKIPEDIA.ORG/WIKI/TEXAS_INSTRUMENTS

TI ha introdotto il computer di casa TI 99 / 4A nel 1981, come una riprogettazione al computer 99/4. È qui rappresentato con periferiche, moduli software plug-in plastici (in basso a destra) e monitor TV a colori. Nel 1981, questa era una grafica all'avanguardia nei display di computer domestici, resa possibile dal microprocessore 9900 a 16 bit di TI.

Vite infinite con il Commodore 64

di Marco Pistorio

A chi non e' rivolto questo articolo:

Questo articolo non è rivolto a chi pensa che giocare con il C64 sia inutile, non è rivolto a chi gioca a vite infinite solo se trova il "loader" del gioco con l'opzione già prevista da chi lo ha crackkato, a chi ritiene sia insensato impegnarsi per imparare qualcosa di più relativamente al C64 perchè non gli cambierà di certo la vita, ed infine non è rivolto a chi conosce bene il linguaggio macchina del C64 (questi ultimi non hanno bisogno del tutorial, sanno già cosa fare).

A chi e' rivolto questo articolo:

Questo articolo è rivolto a chi vuole imparare qualcosa di nuovo relativamente al C64, a chi vuole imparare una tecnica furba per giocare a vite infinite evitando di decompilare un intero gioco, a chi non conosce il linguaggio macchina del C64 o non conosce il linguaggio macchina del C64 in maniera approfondita ma vuole capire quantomeno di cosa si tratta, ed in generale a chi desidera completare un certo gioco, magari sfruttando appunto le vite infinite e non riesce a farlo.

Cosa serve

- PC con emulatore C64 VICE (non è importante utilizzare un release specifico)
- Immagine contenente il gioco (disco o cassetta, è preferibile NON utilizzare immagine cartuccia)
- Programma ad-hoc per confrontare il contenuto di due diversi files byte per byte, ovvero carattere per carattere
- 1 cervello "pronto"!

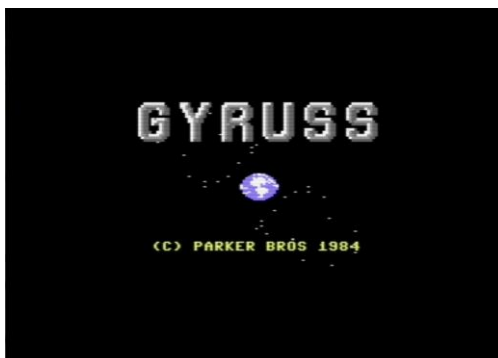
Esposizione delle tecnica

Questa tecnica la imparai diversi anni fa, giocando con il mio fido Commodore64. Per farla funzionare però, Vi illustrerò cosa dovete fare con i mezzi a disposizione oggi, non perchè non sia possibile utilizzare solo il C64 ma perchè è molto più semplice utilizzare allo scopo l'emulatore VICE con il suo programma monitor già previsto al suo interno.

L'idea di base è questa.

Nel C64 abbiamo a disposizione 64 KBytes di RAM. Oggi sembrano incredibilmente pochi, lo so. C'è da dire che non tutta questa RAM è utilizzabile, oltretutto...

Durante l'esecuzione di un gioco, i dati relativi al gioco stesso devono essere caricati nella memoria del Commodore e, quasi sicuramente, insieme agli sprites, alla musica, alla grafica, ci sarà anche l'informazione delle vite a disposizione del giocatore durante la singola partita.

Tale numero verrà probabilmente impostato ad un certo valore, e poi, quando la vita viene persa, tale valore verrà decrementato di una unità. Al raggiungimento del valore o la partita finirà. Questo discorso sarà valido per molti giochi probabilmente ma non per tutti.

Ci saranno giochi ad esempio dove la vita è semplicemente una, ed il gioco procede passo dopo passo, finchè non si commette un qualsiasi grave errore e lì termina il gioco.

Tale meccanismo di gioco NON RIENTRA nella tecnica che vi illustro oggi.

Immaginiamo adesso di giocare ad un qualsiasi gioco avendo ad esempio 4 vite a disposizione.

Facciamo una 'istantanea' di tutto quello che c'è in RAM nel C64 in quel preciso istante.

Perdiamo una delle 4 vite a disposizione.

Facciamo una nuova istantanea di quello che risiede in RAM adesso.

Confrontiamo quindi le due foto.

Cosa otterremo? Probabilmente la stragrande maggioranza dei bytes delle due immagini sarà identica. Perchè? Perchè la musica, gli sprites, gli elementi grafici ed altro ancora sarà sempre memorizzato allo stesso modo.

Ma ci saranno alcuni bytes che differiranno poco tra loro. A noi interessano quelli che differiscono tra loro di una unità (1).

Uno di questi bytes (se non l'unico magari, se siete fortunati...) conterrà proprio l'informazione che ci interessa, ovvero il numero corrente di vite a disposizione.

Perchè? Perchè se io leggevo da qualche parte 4 vite ed ora ne leggo 3, significa che da qualche parte nella RAM tale informazione è conservata!

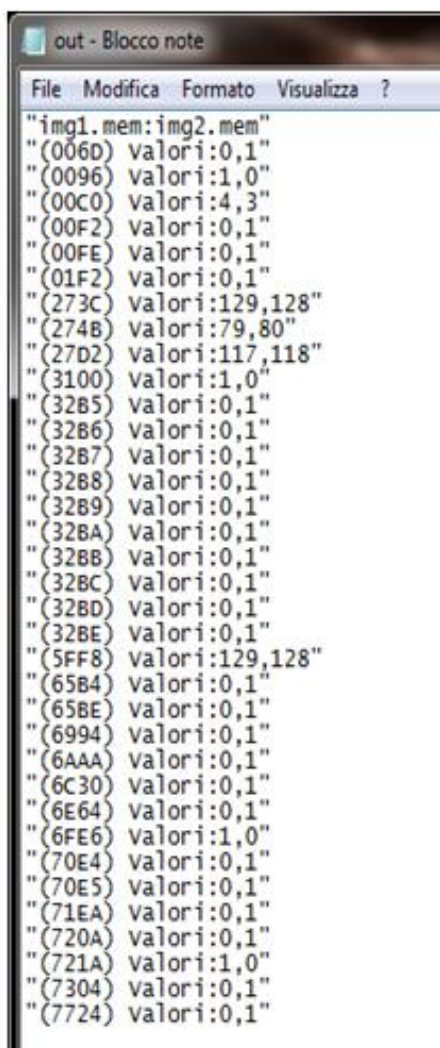
Se avete compreso tutto quello che ho scritto finora, siete a cavallo. Se non dovesse essere così, rileggete nuovamente fino a questo punto.

Spero di essere stato sufficientemente chiaro fin quì.

Individuare la locazione di memoria dove si trova il numero delle vite a disposizione nel gioco è il primo passo per poter giocare a vite infinite.

Il secondo passo è non far decrementare il valore di quella locazione di memoria, oppure reimpostarne manualmente il valore ogni volta.

La tecnica a questo punto dovrebbe esserVi chiara.

Adesso direi di fare un esempio pratico concreto.

Esempio pratico

Consideriamo un qualsiasi gioco dove si procede finchè non si terminano le vite.

Allo scopo prendiamo in esame il gioco **GYRUSS** della Parker Bros anno 1984 (NO CARTUCCIA)

Consiglio una qualsiasi versione senza loaders, più conforme possibile al gioco originale, senza modifiche già apportate da crackers vari, che Vi renderebbero il lavoro ben più difficile.

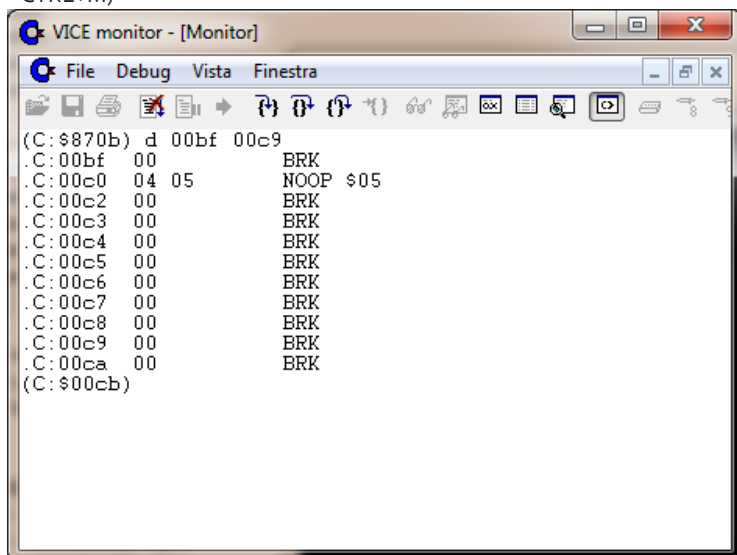
Facciamo partire il gioco. Si tratta di un gioco spaziale, gioco che giocai per la prima volta in sala giochi appena un paio di anni fa (bugia....diciamo una trentina di anni fa, anno più, anno meno)

```
"img1.mem:img2.mem"
"(006D) Valori:0,1"
"(0096) Valori:1,0"
"(00C0) Valori:4,3"
"(00F2) Valori:0,1"
"(00FE) Valori:0,1"
"(01F2) Valori:0,1"
"(273C) Valori:129,128"
"(274B) Valori:79,80"
"(27D2) Valori:117,118"
"(3100) Valori:1,0"
"(32B5) Valori:0,1"
"(32B6) Valori:0,1"
"(32B7) Valori:0,1"
"(32B8) Valori:0,1"
"(32B9) Valori:0,1"
"(32BA) Valori:0,1"
"(32BB) Valori:0,1"
"(32BC) Valori:0,1"
"(32BD) Valori:0,1"
"(32BE) Valori:0,1"
"(5FF8) Valori:129,128"
"(65B4) Valori:0,1"
"(65BE) Valori:0,1"
"(6994) Valori:0,1"
"(6AAA) Valori:0,1"
"(6C30) Valori:0,1"
"(6E64) Valori:0,1"
"(6FE6) Valori:1,0"
"(70E4) Valori:0,1"
"(70E5) Valori:0,1"
"(71EA) Valori:0,1"
"(720A) Valori:0,1"
"(721A) Valori:1,0"
"(7304) Valori:0,1"
"(7724) Valori:0,1"
```

Fig. 1

Il gioco inizia con 5 vite a disposizione, di cui se ne utilizza subito la prima.

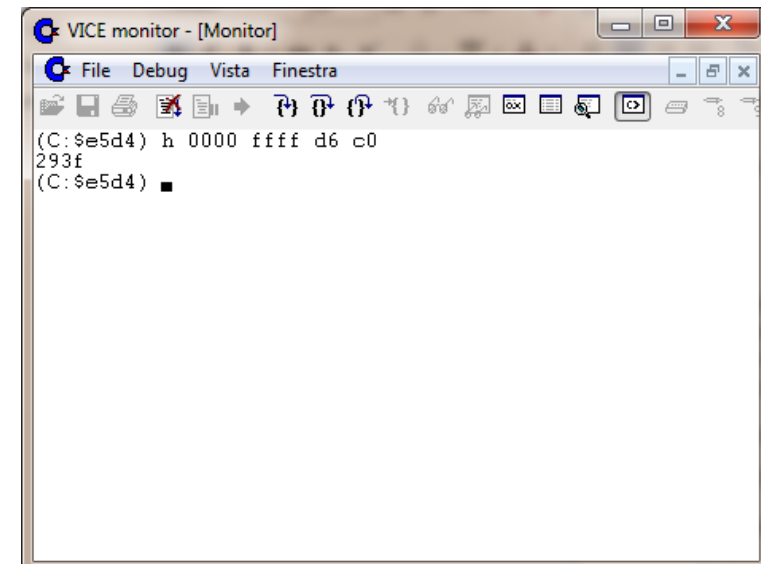
Appena partito il gioco, andiamo sul menù File di VICE e scegliamo Monitor (oppure CTRL+M)

```
VICE monitor - [Monitor]
File  Debug  Vista  Finestra
(C:$870b) d 00bf 00c9
.C:00bf  00         BRK
.C:00c0  04 05      NOOP $05
.C:00c2  00         BRK
.C:00c3  00         BRK
.C:00c4  00         BRK
.C:00c5  00         BRK
.C:00c6  00         BRK
.C:00c7  00         BRK
.C:00c8  00         BRK
.C:00c9  00         BRK
.C:00ca  00         BRK
(C:$00cb)
```

Fig.2

Si aprirà una finestra bianca con un codice strano ed un simbolo di due punti finale.

Fig.3

Questo è il monitor integrato all'interno di VICE. A cosa serve? Ad analizzare il contenuto della memoria del C64, a tradurne il contenuto delle varie locazioni in codice assembly e prevede altre utili funzionalità che in parte utilizzeremo tra non molto.

1. Creiamo la prima immagine di memoria 'campione' digitando il seguente comando

s "img1.mem" 0 0000 ffff

Cosa significa tale comando? Stiamo registrando nel filesystem (device 0) all'interno del file img1.mem il contenuto della memoria compresa tra la locazione 0000 e ffff.

Tali valori sono in formato numerico esadecimale, una notazione molto usata in assembly. I corrispondenti valori decimali di **0000** e **FFFF** sono **0** e **65535**. In parole povere stiamo copiando tutto il contenuto della RAM del C64 in quell'istante dentro il file img1.mem che verrà creato dentro la cartella VICE ove è contenuto l'eseguibile dell'emulatore.

2. Chiudiamo con la X la finestra monitor e continuiamo il gioco

3. Dopo aver perso una vita, ripetiamo la stessa operazione creando un nuovo file di immagine con il comando

S "img2.mem" 0 0000 ffff

A questo punto, basta mettere a confronto i due files ottenuti, byte per byte e cercare di interpretarne correttamente il risultato.

Per fare questo, ho predisposto un apposito programma, scritto in VB6 di cui posso fornirvi sia il codice sorgente che quello eseguibile, che fa l'operazione di confronto automaticamente per voi.

Se volete sfruttare altri software analoghi o volete scriverne uno per conto vostro, siete liberissimi di farlo comunque.

Il programma di confronto, una volta partito, chiede all'interno di 3 caselle di testo, il nome del file da esaminare (in questo caso img1.mem), il secondo file da confrontare (in questo caso img2.mem) ed infine il nome del file dove scrivere il risultato del confronto (qui va bene un nome qualsiasi, ho scelto out.txt)

N.B. Se non specificate percorsi particolari, i files vengono cercati DENTRO LA STESSA CARTELLA OVE OPERA L'ESEGUIBILE!

Dopo aver riempito le 3 caselle di testo ed aver generato il file del confronto, out.txt, otteniamo un risultato come questo (vedi figura 1):

Non preoccupatevi, è molto più semplice di quanto possa sembrarVi a prima vista

Notate il valore tra parentesi tonde, che è un indirizzo in notazione esadecimale, e la scritta Valori, con due valori riportati subito dopo separati da virgola.

Cosa significano? Che ad esempio, nei due file di immagine esaminati, nella locazione di memoria 006D era presente prima un valore 0 e poi un valore 1. E così per tutte le righe di seguito.

Notate qualcosa di interessante? Immagino di sì.

La parte interessante di tutto questo discorso è la seguente riga:

"(00C0) Valori:4,3"

Vi ricordate che il gioco parte con 5 vite di cui una subito in uso?

Poi ne abbiamo persa una, e questa è la locazione che molto probabilmente tiene il conteggio delle vite a disposizione nel gioco.

Abbiamo ottenuto questa utile informazione SENZA ESAMINARE IL CODICE DEL GIOCO, SENZA DECOMPILARE LE ISTRUZIONI DEL GIOCO UNA PER UNA, NON SFRUTTANDO CONOSCENZE DI ASSEMBLY ma semplicemente confrontando due immagini di memoria. Notevole, vero?

A questo punto, cosa è necessario fare? Evitare che il contenuto della locazione 00C0 vada decrementato ogni volta che perdiamo una vita.

Come ottenere tale risultato?

Qui abbiamo tre possibilità, o ci andiamo a mettere un valore molto alto dentro (così quantomeno avremmo a disposizione più di 5 vite) , o alla perdita della vita ci andiamo a rimettere dentro manualmente lo stesso numero dentro, oppure dobbiamo imparare ad usare il monitor del VICE per modificare leggermente il gioco.

Direi di provare direttamente quest'ultima possibilità.

Per andare avanti dobbiamo cercare l'istruzione assembly che varia il contenuto della locazione 00C0.

Notate che le prime due cifre di questa locazione di memoria sono '00'. Tale condizione è importante perché il codice della istruzione che lavora con indirizzi che iniziano per '00', detti indirizzi a pagina 0, è leggermente diverso (ed è anche più veloce in termini di cicli macchina) rispetto alle stesse istruzioni che non operano in pagina 0.

L'istruzione assembly che cerchiamo sarà del tipo: DEC $C0 oppure DEC $C0,X (la parte iniziale 00 viene omessa, per indirizzi in pagina 0)

Perché due possibilità? Perché potrebbe esserci una istruzione che lavora semplicemente all'indirizzo C0, ma potrebbe anche esserci una istruzione che decrementa quello che c'è all'indirizzo C0 e magari quello che c'è all'indirizzo successivo. Tenete presente che il gioco prevede 2 giocatori possibili, ed in effetti esaminando la locazione successiva alla 00C0 ovvero la 00C1 (vedi fig.2), troviamo un valore 5, che è molto probabilmente il numero delle vite del secondo giocatore.

Per vedere tutto ciò, come illustrato anche in figura 2, occorre digitare:

d 00bf 00c9

quindi dare invio.

Tale comando ci permette di 'sbirciare' all'interno di un intervallo di celle di memoria, per cercare di capire cosa c'è all'interno di tali locazioni di memoria.

Questa operazione è nota come 'disassemblare', da qui l'uso della lettera 'd' per lanciare tale funzione.

A questo punto diventa chiaro il fatto che se ci sono due locazioni contigue di memoria che contengono le vite dei due giocatori (ovvero le locazioni 00C0 e 00C1), e se c'è una routine di decremento, sarà la stessa sia che giochi il primo sia che giochi il secondo giocatore. Quindi utilizzerà proprio una istruzione DEC $C0,X e dentro X ci sarà 0 oppure 1 in funzione che a giocare sia il primo o il secondo player.

```
VICE monitor - [Monitor]
File  Debug  Vista  Finestra
(C:$e5d4) h 0000 ffff d6 c0
293f
(C:$e5d4) a 293f nop
.2940  nop
.2941
(C:$2941) d 293f 2949
.C:293f  EA        NOP
.C:2940  EA        NOP
.C:2941  A9 00     LDA #$00
.C:2943  85 ED     STA $ED
.C:2945  4C 00 A7  JMP $A700
.C:2948  A5 EC     LDA $EC
.C:294a  D0 13     BNE $295F
(C:$294c)
```

Fig.4

La logica, quantomeno, ci porta ad immaginare uno scenario simile.

Come fare a scovare dove si trova una istruzione simile nella memoria del C64?

Torniamo al monitor del VICE e vediamo se può aiutarci in questa ricerca.

Apriamo il monitor e digitiamo questa 'magica' istruzione, come illustrato in fig.3:

h 0000 ffff d6 c0

il monitor ci risponderà con:

293f

Abbiamo quasi terminato

Cosa abbiamo fatto? Abbiamo fatto una ricerca (h -> hunt) nell'intervallo 0000-ffff della coppia di valori D6 C0 (che corrispondono alla istruzione di decremento DEC $C0,X a partire dalla locazione di memoria C0 puntata da X)

Come ho fatto a sapere che DEC nn,X si traduce con il valore esadecimale D6?

Perché le istruzioni assembly sono codificate in maniera ben precisa, e consultando la tabella in fondo all'articolo (Tab. 1) ho verificato che l'istruzione DEC nn,X assoluta (cioè che lavora esattamente su un certo indirizzo di memoria), in pagina 0 e con l'uso dell'indice X corrisponde al codice esadecimale C6.

Adesso sappiamo anche a quale indirizzo di memoria tale decremento viene effettuato. Ce lo ha detto il monitor di VICE, ovvero si tratta dell'indirizzo 293f.

Cosa facciamo adesso? Semplice, togliamo l'istruzione.

Come? Beh, non possiamo cancellarla con la gomma, questo è sicuro.

Possiamo però rimpiazzare l'istruzione DEC $C0,X con una coppia di istruzioni NOP.

Perché una coppia? Perché le locazioni da riempire sono due, una contiene C6 e la successiva contiene C0.

L'istruzione NOP è una istruzione speciale, che serve…a non far NIENTE!

Sempre dal monitor, digitiamo:

a 293f nop (e tasto invio)

nop (tasto invio)

infine terzo invio senza battere nulla,come mostrato in fig.4.

(Il comando dato di seguito nella fig.4, d 293f 2949, serve solo per verificare che i due codici operativi NOP siano stati inseriti correttamente)

Il gioco è fatto, chiudiamo il monitor e giochiamo a Gyruss a vite infinite

ATTENZIONE

Nel caso in cui vogliate registrare una copia del gioco con la modifica per le vite infinite, Vi invito a fare innanzitutto una copia del file originale, non si sa mai.

Poi procedere come segue:

a. Determinare le istruzioni che variano l'informazione delle vite restanti
b. Segnarsi da qualche parte tali indirizzi, anche su carta
c. Rimpiazzare tali istruzioni con il codice operativo NOP, come illustrato già in questo tutorial, vedi fig. 4
d. Testare il gioco con tali istruzioni rimpiazzate
e. Se tutto è ok, caricate una copia originale del gioco in memoria, con la classica istruzione LOAD.
f. NON fate partire il gioco con il comando RUN
g. Entrate nel Monitor di VICE (CTRL+M)
h. Cercate nuovamente le istruzioni che variano l'informazione delle vite restanti (dovrebbero coincidere con gli indirizzi segnati precedentemente)
i. Rimpiazzate nuovamente tali istruzioni con codici operativi NOP
j. Uscire dal Monitor di VICE e salvare il gioco, magari con nome diverso, esempio: save "gyruss.vinf",8

Se avete operato correttamente, vi sarà possibile giocare a vite infinite con "Gyruss" facendo partire la seconda versione "gyruss.vinf" che avete opportunamente modificato e salvato. YEAHHHH!!!

LIMITI DELLA TECNICA

Limiti ne esistono. Diciamo che per un buon 80% dei casi, l'istruzione DEC $nn oppure la DEC $nn,X viene utilizzata per lo scopo che vi ho esposto, sia con indirizzi in pagina o che con indirizzi non in pagina 0. Cambieranno solo i valori da ricercare, in funzione della locazione e del fatto che tale locazione si trovi in pagina 0 oppure no.

Ci potranno essere però dei casi in cui il programmatore ha scelto altre vie per decrementare la locazione delle vite. Ad esempio potrebbe scegliere di caricare il dato della vita in uno speciale registro chiamato "accumulatore", decrementare l'accumulatore e registrare sulla locazione di memoria che contiene il numero di vite restanti il risultato finale. Tale discorso si potrebbe verificare anche se il programmatore usasse uno dei registri di puntamento, i registri X ed Y, che potrebbero essere poi decrementati con le apposite istruzioni DEX e DEY ed infine il dato risultante potrebbe essere scaricato nella locazione di memoria delle vite.

In questi casi la tecnica esposta Vi farebbe trovare la locazione delle vite ma per individuare l'istruzione di decremento, la ricerca dovrebbe essere più articolata e richiederebbe una conoscenza maggiore dell'assembly, quantomeno relativa a quanto vi ho accennato in questo paragrafo (uso dell'accumulatore, dei registri indice X ed Y, incrementi e decrementi di tali registri e loro salvataggio in memoria). La ricerca, una volta individuata la locazione di memoria delle vite del giocatore, non si limiterebbe all'istruzione DEC ma si dovrebbero cercare tutte le istruzioni di caricamento LDA, LDX, LDY che fanno riferimento a tale locazione (oppure le istruzioni di scaricamento, rispettivamente STA, STX,STY) e si dovrebbe fare una analisi di ciò che avviene nell'ambito dove tali istruzioni vengono trovate. (Per fortuna nell'assembly del C64 abbiamo solo questi tre registri!)

Un caso invece in cui la tecnica esposta fallirebbe completamente è se il programmatore scegliesse un metodo particolare per memorizzare il dato delle vite.

Vi faccio un esempio:

Considerate un numero ad 8 bit, con un solo bit ad 1 e gli altri a 0. Immaginate che la posizione del bit "1" sia direttamente legata al numero di vite a disposizione, secondo la relazione "numero di vite n = 2 ^ (n-1)", come da schema seguente:

Codice Hex	Modo indirizzamento	Assembler format	Lunghezza Bytes	Numero di Cicli Macchina
CE	Assoluto	DEC nnnn	3	6
DE	Assoluto, X	DEC nnnn,X	3	7
C6	Pagina Zero	DEC nn	2	5
D6	Pagina Zero,X	DEC nn,X	2	6

Tabella 1

Vite n	Valore in binario 2^(n-1)	Valore in decimale 2^(n-1)
8	10000000	128
7	01000000	64
6	00100000	32
...	....	...
1	00000001	1
0	00000000	0

Sebbene possa sembrare un po' machiavellico, una cosa del genere si potrebbe comodamente gestire in assembly facendo i cosiddetti scorrimenti dei registri.

Se la scelta del programmatore fosse questa, il programma che vi ho fornito, che cerca differenze di 1 unità tra due immagini di RAM, fallirebbe miseramente non individuando la locazione di memoria dove tale valore possa essere scaricato tra uno scorrimento ed il successivo, dal momento che tra un passaggio e l'altro la differenza potrebbe essere di 64, 32,16 unità e così via.

Servirebbe quindi un maggiore approfondimento (leggasi anche 'studio') del codice del gioco.

Conclusioni finali

Complimenti per essere arrivati alla fine di questo tutorial !!!

Concludo dicendoVi che questa tecnica si rivela buona in tanti casi ma non in tutti, e se affinate le Vostre conoscenze di assembly, tale tecnica si rivelerà sempre più precisa.

Spero che questo articolo sia stato di Vostro gradimento.

AVVISO IMPORTANTE

Al link http://bit.ly/RM03_VBEsempio potete trovare il progetto sorgente (in Visual Basic) e l'eseguibile del programma creato ad-hoc da Marco per confrontare il contenuto di due diversi files byte per byte, ovvero carattere per carattere. Ovviamente potete scrivervi una vostra utility personale oppure ricorrere ad un file compare freeware.

CBM prg Studio - Cross Development tool per C64

di Giorgio Balestrieri

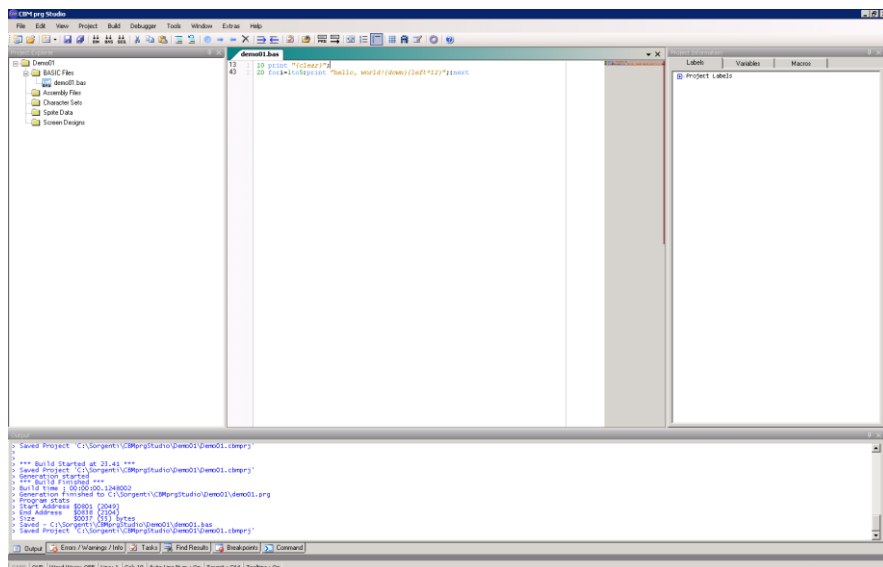



Fig. 1 - CBM prg Studio con un progetto BASIC caricato

Fig. 2 - Creazione nuovo progetto - Passo 1

Nell'articolo "Cross Development tools per C64" presente sul numero 2 di RetroMagazine, è stata presentata una breve ma significativa rassegna degli strumenti "moderni" per il cross development di software per Commodore 64. In questo ed in un prossimo articolo ne approfondiremo uno, il CBM prg Studio, esaminando i tool di cui è composto e sviluppando un piccolo programma di esempio per meglio comprendere le capacità di questo ottimo ambiente integrato dedicato all'immortale macchina di casa Commodore.

Struttura dell'ambiente di sviluppo

Il CBM prg Studio si propone come un ambiente integrato strizzando l'occhio, già dal titolo, al ben noto Visual Studio. Anche se siamo lontani dalle capacità e complessità del prodotto di casa Microsoft, questo software ne racchiude lo spirito, fornendo un IDE capace di gestire e legare insieme tutti i diversi tipi di risorse necessarie per lo sviluppo di programmi per i computer Commodore ad 8bit, mettendo a disposizione gli strumenti per operare su ognuno di essi senza dover ricorrere a prodotti di terze parti. In questo modo avremo il non trascurabile vantaggio di poter gestire un intero progetto con un solo tool.

In questo articolo useremo il Commodore 64 come piattaforma "target" di riferimento, mostrando come si crea un nuovo progetto e come funzionano i seguenti componenti:

- Editor BASIC
- Editor Assembly
- Sprite Editor
- Character Editor
- Screen Editor
- SID Editor
- Screen Code Builder
- Debugger integrato

Creare un nuovo progetto

Iniziare a lavorare con CBM prg Studio è molto semplice. Dal menù "file" ci basterà scegliere "new project" per aprire un wizard che ci guiderà nella creazione del nostro progetto. In "file" sono presenti anche altre due opzioni, "create new BASIC project" e "create new assembly project", che altro non fanno se non creare un nuovo progetto Assembly o BASIC aggiungendo i file di risorse necessari a seconda del tipo di progetto. Seguendo il wizard, dovremo scegliere il sistema su cui far girare il software prodotto (fig.2), indicare il nome progetto ed il nome del file .prg che conterrà il codice compilato, la cartella di destinazione del tutto e opzionalmente, un commento descrittivo del progetto ed il nome del suo autore (fig.3). A questo punto, un click su "next" ed uno su "create" completerà l'opera, chiudendo il wizard e facendo apparire nella parte sinistra dell'IDE un prospetto come quello in fig.1, da dove potremo aggiungere i file per il codice BASIC o Assembly e di risorse come sprite o set di caratteri ridefiniti, scegliendo la voce "new item" dal menù "project", o più semplicemente cliccando con il tasto destro sul ramo di interesse nella vista ad albero del progetto e creare un nuovo file. Ad esempio, per un nuovo file di codice BASIC, è

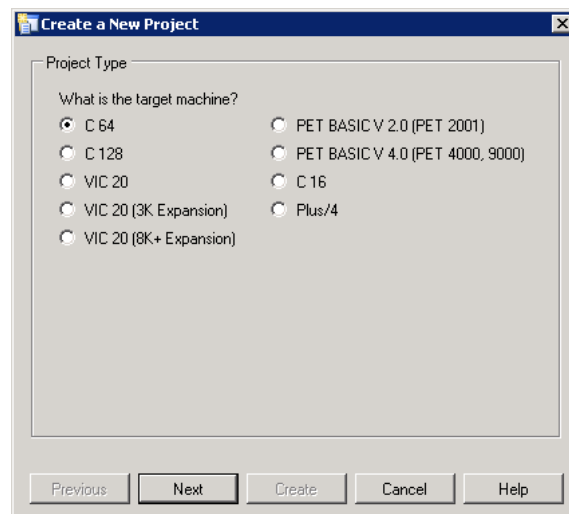
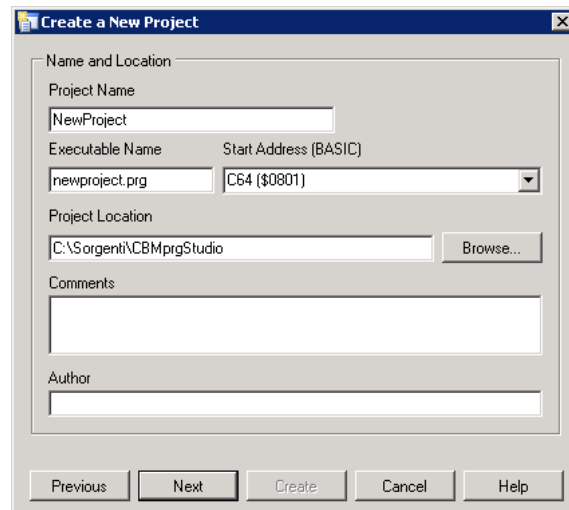



Fig. 3 - Creazione nuovo progetto - Passo 2

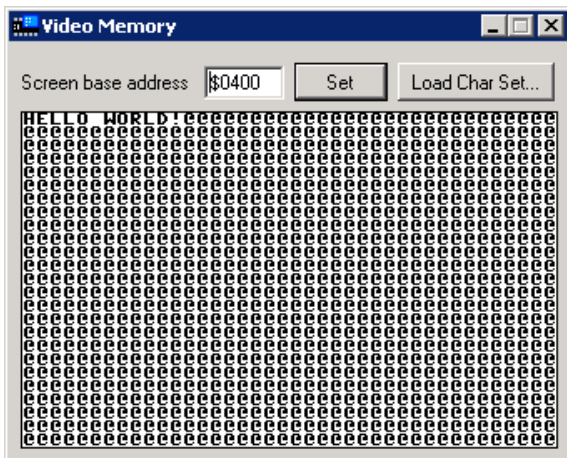



Fig. 4 – Stringa editata con Screen Code Builder

sufficiente cliccare su “BASIC files” con il tasto destro del mouse e scegliere “add new file”, indicando poi il nome da dare al file che stiamo creando e confermare con “Ok” per aggiungerlo al progetto. In questo modo si aprirà una finestra di editing al centro dell’IDE per iniziare subito a scrivere il nostro codice. Il procedimento è del tutto analogo per gli altri tipi di risorse.

Editor BASIC

Chi ha scritto programmi in BASIC sul Commodore 64 ai bei tempi, ricorderà la flessibilità e la semplicità d’uso dell’editor incluso nel sistema operativo della macchina. Era talmente semplice e comodo da usare da far sembrare goffi e farraginosi gli editor della maggior parte dei micro computer concorrenti. Questo negli anni ‘80. Se vi venisse in mente di provare a scrivere codice BASIC sul C64 oggi, probabilmente vi ritrovereste a prendere a testate la prima superficie verticale disponibile dopo soli 5 minuti. Per fortuna esiste il CBM prg Studio, che offre un eccellente editor moderno con integrate utili funzioni per la scrittura di sorgenti in BASIC, come ad esempio la gestione automatica dei numeri di riga. Premendo invio alla fine di una riga di istruzioni, l’editor provvederà a generare un numero di riga valido per proseguire la scrittura del programma. Questo avviene anche se si preme invio al termine di una riga nel mezzo del listato: l’editor sceglierà per noi un numero di riga compreso tra quello corrente e quello della riga successiva.

Un’altra importante caratteristica, è la rinumerazione delle righe. Se dovesse essere necessario spostare un pezzo di codice in un’altra posizione del listato, invece di cambiare a mano tutti i numeri di riga, possiamo usare la funzione “renumber” presente nel menù “tool” per risistemare tutto in automatico, compresi i numeri di riga delle eventuali istruzioni goto e gosub presenti. Notevole poi è la possibilità di definire regioni di codice a cui applicare il “code folding”, cioè ripiegare una parte del codice in una sola linea, escludendolo dalla vista globale del listato, molto utile per mantenere il colpo d’occhio su sorgenti particolarmente lunghi e per suddividere il codice in blocchi funzionali facilmente individuabili. Inoltre, poichè CBM prg Studio provvede a compilare il codice BASIC in op-codes, avremo l’enorme vantaggio di poter scrivere tranquillamente righe più lunghe di 80 caratteri; l’IDE ci avvertirà se superiamo questo limite (che è proprio dell’editor del

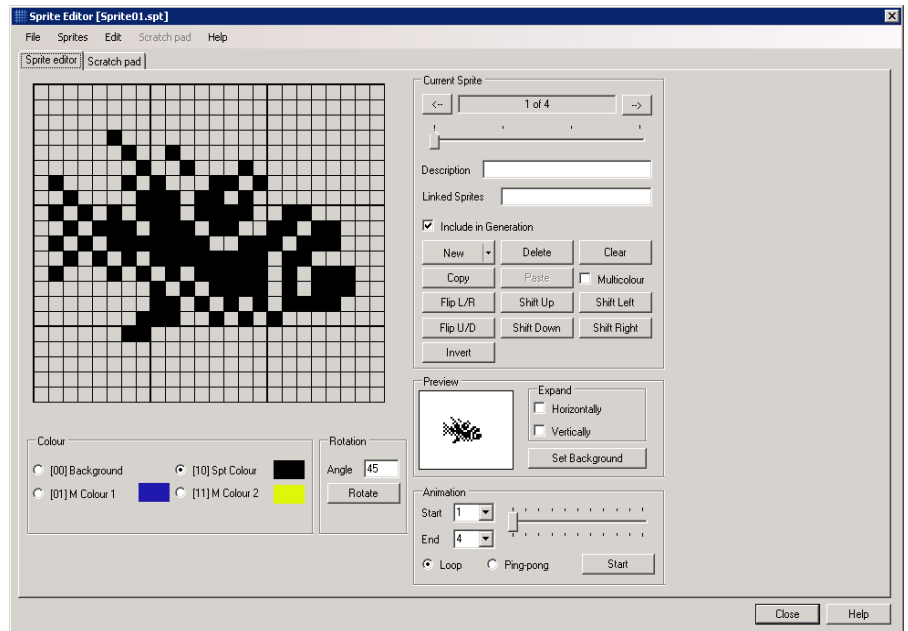



Fig. 5 - Sprite Editor

C64), ma non se ne farà un grosso problema, ci ricorderà semplicemente che righe così lunghe non potremo poi modificarle su un Commodore 64 (originale o emulato che sia).

Editor Assembly

Questo strumento, come suggerisce il nome, si attiva quando apriamo un file .asm. La qualità e le caratteristiche sono le stesse dell’editor BASIC (di fatto è lo stesso editor riadattato al codice assembly) per cui piuttosto che rielencare, preferiamo scendere nel dettaglio delle capacità dell’assemblatore che riceve il codice scritto con questo tool. Grazie alla possibilità di

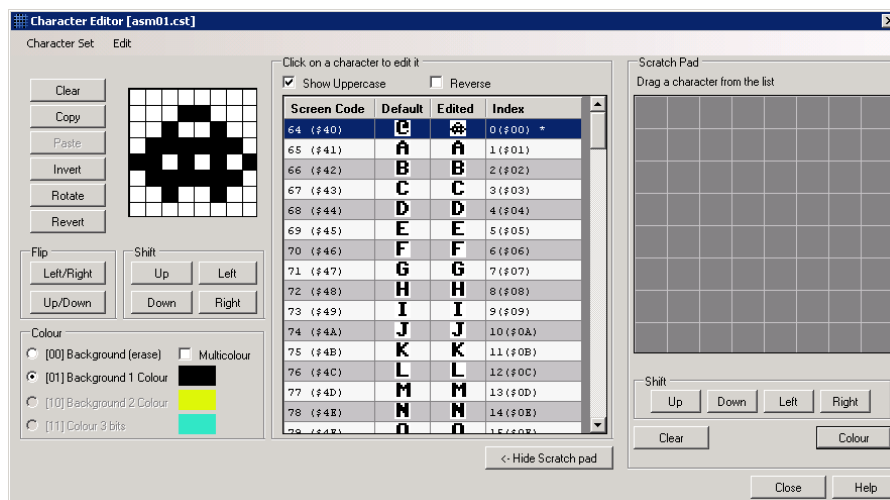



Fig. 6 - Character Editor

utilizzare label come riferimento delle istruzioni di salto, qui non avremo numeri di riga (o più precisamente, locazioni di memoria) da gestire; possiamo scrivere le nostre istruzioni una dopo l'altra utilizzando una label quando serve. Per indicare la locazione di inizio del nostro codice, possiamo utilizzare la direttiva *=$<locazione> che l'assemblatore utilizzerà per sapere dove collocare il codice in linguaggio macchina prodotto, provvedendo a tradurre in locazioni di memoria le label definite. Una interessante caratteristica è che quando ci troviamo a lavorare con un progetto misto BASIC/Assembly, possiamo rendere visibile una label definita nell'Assembly anche al BASIC; in questo modo per richiamare una subroutine in LM dal BASIC sarà sufficiente scrivere sys <label> invece della sua locazione di memoria. Grazie a questo tipo di gestione, la rilocazione del codice diventa un'operazione assolutamente banale; basta indicare una nuova locazione con la direttiva *= e ripetere il processo di build. Il compilatore penserà a tutto, riassegnando i valori delle label definite. Per conoscere i byte alto e basso di una label, utili in casi come le operazioni che coinvolgono l'indirizzamento indiretto, occorre utilizzare gli operatori < e >. Ad esempio, per caricare in $fb ed in $fc l'indirizzo di un buffer identificato dalla label _my_buffer, basterà utilizzare le istruzioni mostrate nel riq.1:

```
        lda #<_my_buffer
        sta $fb
        lda #>_my_buffer
        sta $fc
_my_buffer
byte  $30,  $31,  $32,  $33,  $34,  $35,
$36, $37
byte  $38,  $39,  $3A,  $3B,  $3C,  $3D,
$3E, $3F
```

riq.1

e il compilatore provvederà a far quadrare i conti. Sulle label è possibile anche compiere operazioni aritmetiche per accedere a locazioni di memoria prima o dopo quella della label. Ad esempio, facendo riferimento al codice del riq. 1, **lda _my_buffer+10** caricherà il valore $3A nell'accumulatore. Altre interessanti opzioni comprendono direttive per la compilazione condizionale, la definizione di variabili e buffer di memoria e la definizione di macro.

Sprite Editor

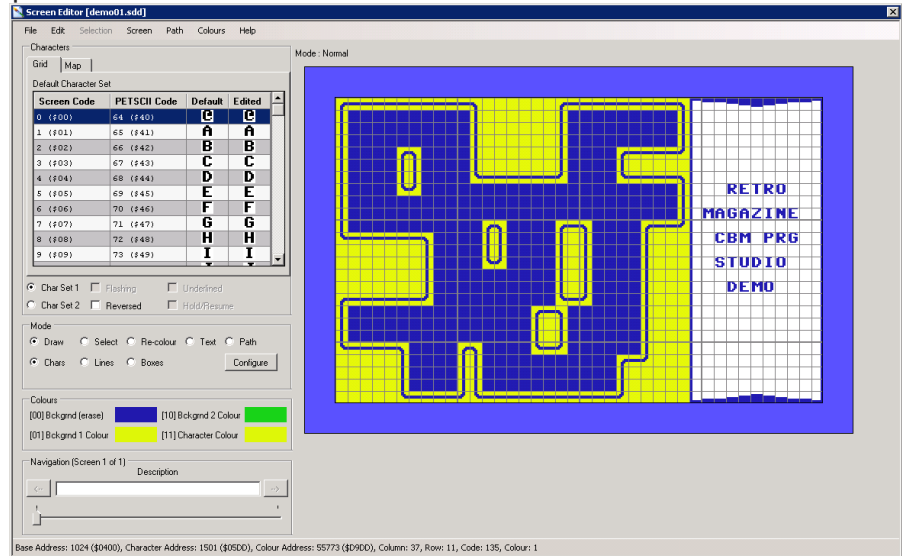



Fig. 7 - Screen Editor

Avete la necessità di aggiungere sprite al vostro progetto? Nessun problema, selezionate con il tasto destro la voce "sprite data" nell'elenco delle risorse e scegliete "add new file" o riutilizzate un file di sprite creato per altri progetti. Quale che sia la vostra decisione, con un doppio click sul nome del file che comparirà sotto "sprite data" si aprirà l'editor per gli sprite, uno strumento piuttosto completo e ben integrato con gli editor BASIC ed Assembly, con potremo disegnare tutti gli sprite monocromatici o multicolore che ci necessitano e di provare le sequenze animate grazie ad un minimale ma utile player. Grazie ad uno scratch pad poi potremo riorganizzare gli sprite disegnati raggruppandoli per crearne di più grandi o sovrapponendoli per creare uno sprite complesso di dimensioni standard. Le operazioni possibili su uno sprite sono il ribaltamento sull'asse orizzontale e verticale, l'inversione dei suoi pixel e lo spostamento di tutti i punti in una delle quattro direzioni cardinali all'interno della griglia di definizione, utile per l'aggiustamento al pixel dei frame animati. Mancano all'appello tool per le primitive grafiche di base, come il riempimento di aree con un colore a scelta e il disegno di linee, cerchi e rettangoli. Un peccato, avrebbero reso perfetto questo pratico editor l'autore deciderà di includerli. Magari in una futura versione. Una volta disegnati, i dati degli sprite possono essere esportati direttamente nei nostri sorgenti BASIC o Assembly in forma di linee DATA nel primo caso, o di buffer di byte nel secondo. E' possibile anche decidere se esportare i dati per tutti gli sprite o solo per una parte di essi, il che viene molto comodo se vogliamo avere una libreria di sprite pronti all'uso in un solo file spt e prendere solo quelli di cui abbiamo bisogno per il nostro progetto. Ultima feature dell'editor, che può tornare utile in alcuni contesti, la capacità di importare sprite a partire da un sorgente BASIC/Assembly o da una bitmap.

Character Editor

Per ridisegnare i caratteri del nostro retro-sistema, questo tool è tutto ciò che ci serve. E' molto simile allo sprite editor, sia come interfaccia che come caratteristiche. Possiamo compiere le stesse operazioni di spostamento e inversioni speculari possibili sugli sprite ed arrangiare i caratteri ridefiniti su uno scratch pad, così da comporre blocchi di più caratteri per formare un unico "disegno". Anche qui abbiamo la possibilità di esportare i dati dei caratteri in un listato BASIC o Assembly e di importarli da varie sorgenti. Come per lo sprite editor, sono assenti tool per le primitive grafiche di disegno, anche se la loro mancanza si fa sentire meno per la ridotta dimensione dello spazio disponibile per disegnare (griglia da 8x8 bit contro 24x21). Per il resto, questo strumento si utilizza in maniera molto simile, se non identica, allo sprite editor. Non è

presente il player per le animazioni, il che è piuttosto comprensibile ma sarebbe stato apprezzabile, ad esempio per lo sviluppo di giochi dove le animazioni con i caratteri possono in effetti essere utilizzate. Ma forse stiamo chiedendo troppo sia all'IDE che al suo autore il quale, lo ricordiamo, ha sviluppato CBM prg Studio tutto da solo.

Screen Editor

Diversamente a quanto indicato nell'articolo di presentazione dei tool per cross-development per C64, in questa suite è presente un editor di schermate (sono assenti quelli di mappe di schermate e livelli). Anche in questo caso, si nota l'elevata qualità e cura impiegata dal programmatore nella sua progettazione. Con esso possiamo disegnare una schermata utilizzando il set di caratteri standard della macchina o uno ridefinito, componendola carattere per carattere, scelti di volta in volta dal una griglia di caratteri o, opzione che personalmente trovo più efficace, da una mappa, oppure tracciando linee e rettangoli (composte da un carattere selezionato) o inserendo righe di testo. Per il disegno di rettangoli, di default questi sono disegnati utilizzando i caratteri semigrafici del C64 con gli angoli di 90°, ma configurando il tool è possibile utilizzare qualunque carattere ci piaccia per tracciarli (ad es., possiamo avere gli angoli arrotondati). E' anche possibile indicare se l'interno deve essere vuoto o riempito con un carattere in particolare (di default è lo spazio); in questo modo possiamo tracciare un riquadro intorno ai caratteri già presenti sullo schermo o un blocco "solido" che li copra. Altre utili opzioni prevedono la sostituzione di un carattere con un altro su tutto lo schermo o di ricolorare una determinata area o singolo carattere cambiandogli il colore in primo piano o di sfondo. Lo strumento permette di disegnare contemporaneamente più schermate, che però vengono viste come separate e non appartenenti ad un'unica mappa, ma è comunque molto utile per creare giochi multiscreen o utility che

Fig. 8 - SID Tool

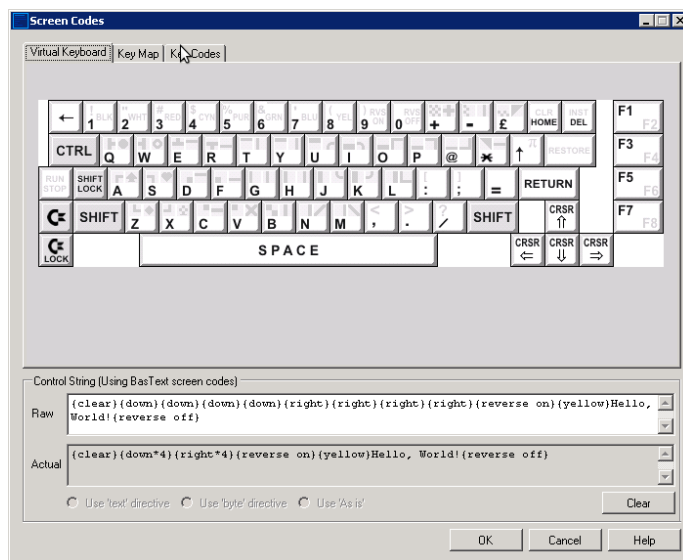
prevedano l'uso di più "pagine" di opzioni e funzionalità. Le schermate così composte potranno poi essere esportate in un listato BASIC, generando automaticamente le istruzioni print necessarie a disegnarle, oppure come blocchi di dati, uno per la mappa di caratteri ed uno per quella di colori, da copiare poi sullo schermo tramite poke o relative istruzioni in Assembly.

Ultima, notevole, caratteristica è la definizione di path per gli sprite. Grazie a questo tool semplicissimo da utilizzare, possiamo tracciare un percorso da far compiere ai nostri sprite attraverso la schermata disegnata. Per farlo, possiamo tracciare i segmenti che compongono il percorso desiderato o utilizzare una funzione matematica che calcolerà i punti sullo schermo che faranno muovere il nostro sprite (non è possibile combinarli, automaticamente almeno). I dati sono poi esportati come un blocchi di istruzioni data per il BASIC o di byte per l'Assembly, ma le routine per utilizzarli e muovere gli sprite dovranno essere scritte da noi. Lo strumento path, può anche essere utilizzato per il disegno della schermata indicando al tool di convertire in caratteri i segmenti del percorso.

Fig. 9 - Screen Code Builder

SID Tool

Con questo strumento, disponibile solo per i progetti destinati a girare su C64, possiamo creare semplici effetti sonori da includere nei nostri programmi. Lo strumento, oltre a permettere di modificare tutti i registri ed attivare tutte le caratteristiche del SID, include anche un emulatore per poter testare il risultato come se fosse "suonato" attraverso il chip hardware. Una tastiera musicale virtuale ci aiuta a testare gli effetti dei nostri esperimenti. Purtroppo l'editor non permette di scrivere brani musicali, è un editor orientato al chip audio, non un tracker musicale ma forse, una volta di più, stiamo chiedendo troppo sia all'IDE che al suo autore...

Dopo aver raggiunto il risultato voluto, possiamo esportare le impostazioni raggiunte come codice per poter replicare l'effetto sonoro nel nostro programma BASIC o Assembly, in maniera analoga a tutti gli altri tool, con in più la possibilità di chiedere al SID Tool di generare per noi il codice necessario al setup del chip audio, così da predisporlo ad eseguire correttamente il nostro suono.

Screen Code Builder

Chi ha programmato in BASIC sul biscottone, apprezzerà moltissimo questo strumento. Ricordate quello che accadeva quando si digitava un doppio apice nell'editor del C64? Si attivava il famoso (o famigerato) "quote mode", in cui i tasti per il posizionamento del cursore, i comandi per il cambio del colore del testo, quelli per pulire lo schermo ed i tasti di funzione venivano soppressi ed invece di svolgere il compito loro assegnato, veniva mostrato un carattere in reverse che avrebbe eseguito l'azione che ci si aspettava quando la stringa fosse stata stampata a video tramite una istruzione print. Il "quote mode" terminava quando si digitava di nuovo un doppio apice, chiudendo così la stringa. Questa modalità di editing, utile ad esempio per pilotare da codice il cursore sullo schermo e cambiare colore al testo stampato, era al tempo stesso un'idea geniale ed una maledizione. Geniale perché permetteva di gestire tutta una serie di operazioni a video senza dover usare istruzioni dedicate ma eseguendole in una forma di primordiale "macro di tastiera", risparmiando tempo e memoria, una maledizione perché una volta attivato il "quote mode", la possibilità di correggere eventuali errori di battitura si riduceva drasticamente poiché non era possibile tornare indietro e modificare le parti errate, ma si doveva cancellare tutto quanto scritto dopo l'errore e ricominciare da lì (si poteva "chiudere" la stringa e muovercisi all'interno liberamente, ma se si dovevano inserire altre "macro", si era costretti ad una serie di virtuosismi non banali).

Questo utilissimo strumento rende tutto questo superato. Tramite una tastiera virtuale, possiamo inserire tutti i "macrocomandi" per posizionare il cursore (ed altro) rapidamente e modificare la stringa come più ci piace, senza i limiti del "quote mode". In più, al posto dei caratteri in reverse che apparivano nell'editor del C64, qui vengono usate delle vere e proprie macro che possono essere copiate ed inserite nelle nostre stringhe. All'atto della compilazione del progetto, i "macrocomandi" verranno sostituiti con i caratteri speciali che il C64 si aspetta. In fig.4 possiamo vedere un esempio di stringa editata con lo Screen Code Builder. Il codice presente nel riquadro "Control String" permette di pulire lo schermo e di stampare la stringa "Hello, World!" in giallo a caratteri invertiti sulla riga quattro a partire dalla quarta colonna. Notare che sono presenti due sezioni, "raw" ed "actual"; nella prima, vediamo l'elenco completo dei macrocomandi digitati, nella seconda, gli stessi in forma "compressa". Possono essere utilizzate entrambe le forme per le nostre stringhe ma ovviamente la forma compressa è da preferire. Le stringhe di macro possono essere anche esportate per l'Assembly; in questo caso possiamo scegliere se utilizzare la direttiva TEXT o quella BYTE per memorizzarle.

Debugging

Per i progetti in Assembly, CBM prg Studio dispone della capacità di effettuare il debug direttamente dall'IDE, usando un debugger interno o le capacità di debbuging di VICE. Qui descriveremo il funzionamento del debugger interno, a cui è stato associato un emulatore di 65xx, un viewer per la memoria ed uno per l'area dello schermo a caratteri. In fig.10 è mostrato il debugger aperto con i due viewer associati. Nella parte alta del debugger viene visualizzato il contenuto dei registri del processore, seguito da un riquadro contenente il codice compilato (notare i riferimenti alle label definite nel codice). Nell'ultimo pannello, quello più in basso, è mostrato il contenuto dello stack, eventuali locazioni di memoria "sotto osservazione" e l'elenco dei breakpoint.

Nel memory viewer viene mostrato il contenuto "raw" della memoria, mentre in video memory possiamo vedere il contenuto dello schermo come se fossimo sulla macchina fisica, con tanto di possibilità di caricare un set di caratteri ridefinito. Notare che di default tutte le locazioni della memoria emulata da CBM prg Studio contengono il valore zero, il che spiega perché la finestra "Video memory" sia riempita dagli "at", ovvero il simbolo "@", comunemente noto come chiocciolina. Il debugger permette di definire dei breakpoint, punti in cui interrompere l'esecuzione del codice, e di mostrare in ogni istante il contenuto di qualsivoglia locazione di memoria ci interessi, aggiungendola alla watch list. E' anche possibile scegliere la velocità con cui far eseguire il codice dal debugger o di disattivare l'aggiornamento in tempo reale del contenuto dei registri, della watch list e dell'indicatore del program counter. Nel menù troviamo poi degli utili strumenti per alterare il contenuto dei registri o per inizializzare o copiare zone di memoria. Nel complesso, questo strumento si mostra semplicissimo da usare ma estremamente efficace per testare il nostro codice Assembly; a meno di non dover utilizzare caratteristiche avanzate della macchina, molto del lavoro di debug può essere effettuato con questo tool.

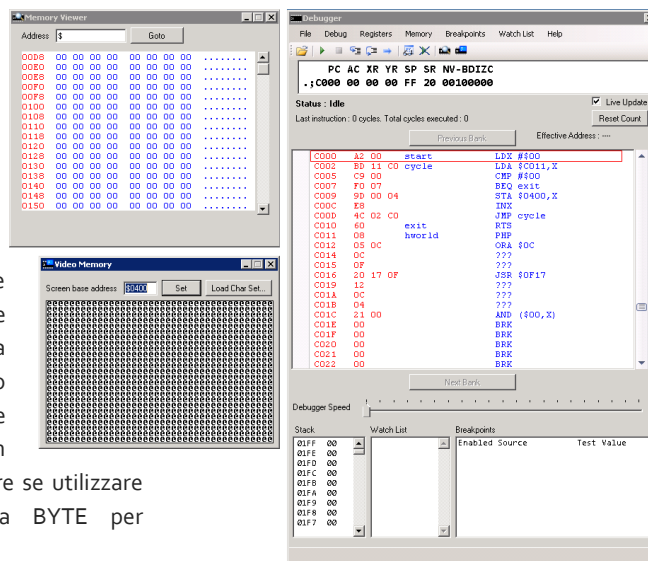



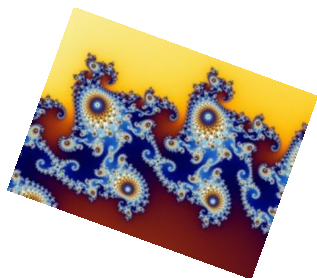
Fig. 10 - Debugger Integrato

Conclusioni

Siamo giunti al termine di questo articolo, con cui speriamo di aver reso giustizia alla complessità e completezza di questo eccellente strumento di sviluppo. In un prossimo numero mostreremo un semplice progetto sviluppato con il CBM prg Studio, così da vedere "dal vivo" come tutte le sue caratteristiche ci aiutino nella creazione di un software per C64. Nella versione 3.11 utilizzata per questo articolo, la più recente disponibile al momento della scrittura, sono stati notati alcuni bug (ad es. non è possibile esportare un singolo sprite o un range di essi da un file .spt) prontamente segnalati all'autore, che ha assicurato di sistemarli nelle prossime versioni di CBM prg Studio.

RetroMath: i Frattali

Alla scoperta delle meravigliose forme “complesse” della natura

di Giuseppe Fedele e Marco Pistorio

Entriamo nell’affascinante mondo dei frattali scoprendo cosa si nasconde dietro le belle immagini che costruiamo al computer e quali strumenti matematici ci servono per studiare queste nuove strutture.

Spieghiamo come definire gli insiemi di Julia e di Mandelbrot e ***costruiamo un programma per visualizzarli su un computer…ovviamente retro!***

Avete mai mangiato un broccolo romano? Questo simpatico cavolo dalla forma piramidale e dalle curiose rosette si sposa perfettamente con una bella porzione di gamberetti in un mix di sapori irresistibile per un primo piatto di fusilli tanto semplice quanto sfizioso.

Figura 1. Broccolo romano

Ma a noi interessa qui il suo fascino matematico: se ne ingrandiamo anche una piccola parte, riproduciamo in scala la stessa figura di partenza, oppure ritroviamo, in scala, caratteristiche strutturali simili. Questa proprietà, legata alla natura irregolare, è detta **autosomiglianza** e implica l’assenza di regolarità. Esistono tante altre forme in natura che presentano questa proprietà di autosomiglianza. Un albero ne è un altro esempio: il tronco si divide in due grossi rami, ognuno dei quali si divide nuovamente in due rami più piccoli e così via, fino a formare l’intera chioma.

Figura 2. Un albero

Nella felce, la disposizione delle foglie sul ramo più grande è identica a quella di ogni foglia piccola laterale (diminuisce solo la dimensione). Anche nel nostro corpo, la ramificazione dei bronchi possiede questa proprietà: il tubicino della trachea si divide un due altri tubicini chiamati bronchi, ognuno dei quali a sua volta si divide nuovamente in due parti sempre più sottili fino a formare l’intera massa dei polmoni.

Figura 3. Felce

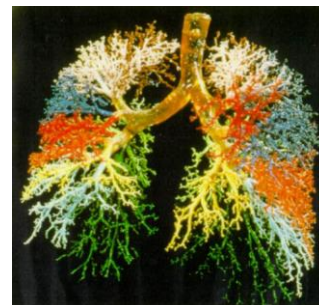



Figura 4. Bronchi

Nel 1975, nel suo libro “Les objets fractals”, Benoit Mandelbrot, matematico polacco, definisce questi oggetti **frattali**: la parola “frattale” deriva dal latino “fractus” che significa appunto spezzato, rotto, fratturato come a sottolineare la natura irregolare della forma.

Nascono quindi i frattali, modelli che racchiudono quelle forme molto complesse che si trovano in natura e che fino ad allora non erano state considerate riproducibili con regole matematiche.

Lo stesso Mandelbrot scriveva infatti che:

“La geometria euclidea è incapace di descrivere la natura nella sua complessità, in quanto si limita a descrivere tutto ciò che è regolare. Tutti gli oggetti che hanno una forma perfettamente sferica, oppure…
mentre osservando la natura vediamo che le montagne non sono dei coni, le nuvole non sono delle sfere, le coste non sono dei cerchi,

ma sono oggetti geometricamente molto complessi".

E mentre gli elementi della geometria (linee, cerchi, triangoli, quadrati,...) si possono visualizzare facilmente, gli oggetti frattali sono processi, algoritmi che possono essere trasformati in forme visibili sono con l'aiuto di un computer.

Sistemi dinamici discreti

Ma prima di entrare nel magico mondo dei frattali è necessario fare una breve introduzione sui **sistemi dinamici**.

Un sistema dinamico discreto (**s.d.d.**) si può considerare come una funzione che genera una successione di stati mediante la sua ripetizione iterativa.
Non spaventatevi, è molto più semplice di quello che pensate!

Esempio 1
Considerate, ad esempio, il seguente s.d.d.:

$$z_{k+1} = z_k + 1$$

dove la variabile z può assumere un qualunque valore reale. Questa formula ci dice che, una volta noto il valore della variabile *z* all'istante *k*, è possibile calcolare il valore della variabile *z* all'istante *k+1* sommando la quantità *1*.
Quindi se conosciamo *z* all'istante iniziale (per convenzione l'istante iniziale è lo zero e il valore della variabile *z*, in tale istante, viene chiamata **condizione iniziale**), siamo in grado di ricostruire tutta la sequenza che, ovviamente, è infinita.
Supponiamo quindi che la condizione iniziale sia $z_0 = 3.5$, allora la sequenza sarà

$$z_0 = 3.5$$
$$z_1 = z_0 + 1 = 4.5$$
$$z_2 = z_1 + 1 = 5.5$$
$$z_3 = z_2 + 1 = 6.5$$
$$\vdots$$

Esempio 2
Facciamo un altro esempio per introdurre degli altri concetti che ci serviranno per disegnare le immagini frattali. Consideriamo quest'altro s.d.d.

$$z_{k+1} = z_k^2.$$

Se applichiamo l'iterazione con la condizione iniziale $z_0 = 2$, otteniamo $z_1 = 4, z_2 = 16, z_3 = 256$ e così di seguito. La sequenza di numeri generata viene chiamata **orbita** del s.d.d. In questo caso l'orbita tende all'infinito perché il valore di *z* cresce, da una iterazione all'altra.
Se invece scegliamo come condizione iniziale $z_0 = 0.5$, l'orbita sarà $z_1 = 0.25, z_2 = 0.0625, z_3 = 0.0039, \ldots$ In questo caso l'orbita tende a zero.

Il punto che l'orbita tende a raggiungere viene chiamato **attrattore**.
Quindi, per la prima sequenza l'attrattore è infinito, nella seconda l'attrattore è lo zero.
E se $z_0 = 1$, quale sarà l'orbita associata? Notiamo che z_k sarà sempre pari ad *1*. In questo caso l'orbita è costituita da un solo punto, guarda caso chiamato **punto fisso**.
Siamo davvero vicini, infatti molte delle immagini frattali che incontriamo spesso sono legate al s.s.d.

$$z_{k+1} = z_k^2 + c$$

dove stavolta z e c sono numeri complessi (vedi la scheda di approfondimento). L'orbita associata al s.d.d. sarà quindi una sequenza di numeri complessi che dipende dalla condizione iniziale e dal valore di c.

L'insieme di Julia

Fissiamo un valore di c e facciamo evolvere il nostro s.d.d. considerando, come condizione iniziale z_0, diversi punti del piano complesso. Al variare di z_0, si possono ottenere orbite che tendono all'infinito (chiameremo questi punti **fuggitivi**) e orbite che continuano a permanere nei dintorni dell'origine (chiameremo questi punti **prigionieri**). L'insieme dei punti prigionieri e l'insieme dei punti fuggitivi sono separati da una frontiera (punti **guardiani**) che prende il nome di **insieme di Julia** in onore del matematico Gaston Julia che, per primo, studiò questo problema.

Come si fa a sapere se un punto è un fuggitivo oppure un prigioniero? La risposta è data da un teorema dimostrato dallo stesso Mandelbrot: se un punto si trova ad una distanza maggiore o uguale a due unità dall'origine, allora è destinato all'infinito; se invece si trova ad una distanza minore a due unità dall'origine, allora è un punto prigioniero.

Per vedere gli insiemi di Julia abbiamo bisogno dell'aiuto di un computer. In teoria, fissato un valore di c, dovremmo dimostrare

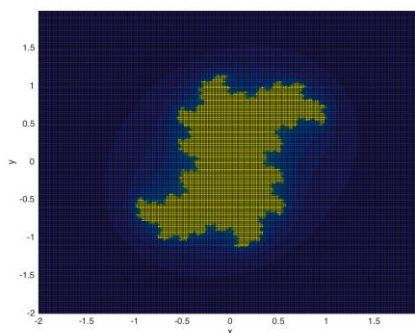



Figura 5. c=0.3-0.4i

per ognuno dei punti del piano se la sua orbita tende a infinito o a zero. Nella pratica ciò è impossibile e per questo utilizziamo un algoritmo da implementare su un computer (rigorosamente retrò): si fissa il numero complesso c e si fa evolvere la successione fino ad un numero fissato di iterazioni o fino a quando il modulo alla k-ma iterazione non supera il valore 2. Ad ogni iterazione si colora con colore diverso i punti nell'intervallo iniziale considerato che sono in modulo minori di 2.

A questo punto appare chiaro che per ciascun valore prefissato di c usato nella formula di iterazione, appare un diverso insieme di Julia, pieno di prigionieri (Figure 5, 6 e 7).

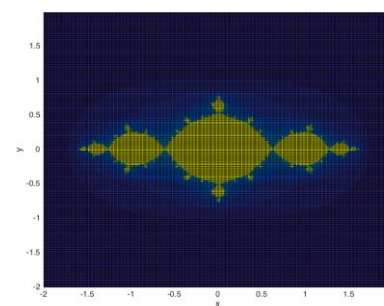



Figura 6. c=-1

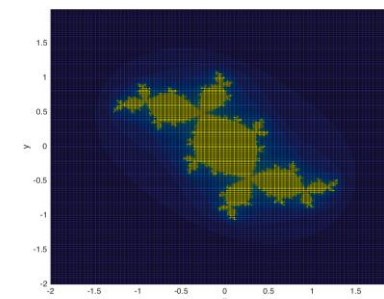



Figura 7. c=-0.122+0.745i

Il codice per disegnare i frattali di Julia su Commodore 128 è mostrato in seguito. In Fig. 8 vedete il risultato ottenuto con $c = -1$.

L'insieme di Mandelbrot

A differenza dell'insieme di Julia, quello di Mandelbrot non fissa il valore dl parametro c ma considera l'insieme dei punti di c per i quali non è divergente l'ormai nota legge ricorsiva con condizione iniziale $z_0 = 0$.

Anche in questo caso, se il modulo del numero complesso z, ad una determinata iterazione, diventa superiore a 2, allora l'orbita associata diverge.

Si fissa dunque un numero massimo di iterazioni, dopo le quali si assume che se il modulo di z non è maggiore di 2 allora il punto considerato fa parte dell'insieme di Mandelbrot.

I bei colori che si vedono associati agli insiemi di Mandelbrot e di Julia sono assegnati in dipendenza del numero di iterazioni che sono necessarie per capire se un punto appartiene o no all'insieme. In questo modo è possibile associare un colore in funzione della velocità con cui l'orbita diverge (ma questa è un'altra storia!!!).

Dimenticavo, se eseguite i codici fatelo all'ora di pranzo, nel frattempo cucinate un bel piatto di fusilli, cavolo romano e gamberetti e dopo date un'occhiata al vostro retro PC per controllare se la computazione è terminata!!!

(Giuseppe e Marco)

Bibliografia:

[1] B. Mandelbrot, *The Fractal Geometry of Nature*. New York: W.H. Freeman and Company, 1982.

[2] B. Mandelbrot, *Les objets fractals*. Champs, Flammarion, 1975.

[3] E. Salinelli, F. Tomarelli, *Modelli Dinamici Discreti*. Springer, 2014.

I numeri complessi

La necessità di introdurre i numeri complessi nasce quando si prova a risolvere l'equazione di secondo grado $x^2 = -1$. Il problema in questo caso è che la funzione radice quadrata non è definita per numeri negativi. Per questo si introduce una nuova grandezza, l'unità immaginaria definita come $i^2 = -1$. Questa nuova grandezza ci permette di concludere che le soluzioni dell'equazione $x^2 = -1$ sono $x = \pm i$.

Un numero complesso è identificabile come un punto di un piano (complesso) ed è comunemente rappresentato in due modi: nella forma cartesiana e nella forma esponenziale. Nella forma cartesiana un numero complesso z viene individuato dalle sue coordinate reali x e y e si può scrivere $z = x + iy$. La coordinata x è la parte reale di z mentre y è la parte immaginaria di z. Nella forma esponenziale il numero complesso z viene invece indicato dal modulo $|z|$, ossia la distanza del punto z dall'origine, e dall'argomento, ossia l'angolo θ compreso tra la direzione positiva dell'asse x e la semiretta uscente dall'origine e passante per z. Il modulo del numero complesso, a partire dalle sue componenti cartesiane, può essere espresso come $|z| = \sqrt{x^2 + y^2}$. Le operazioni tra numeri complessi sono analoghe alle operazioni algebriche tra binomi:

$$(a + ib) + (c + id) = (a + c) + i(b + d)$$

$$(a + ib) * (c + id) = ac + ibc + ida + i^2bd = (ac - bd) + i(ad + bc)$$

L'operazione iterativa, necessaria a calcolare l'insieme di Julia sarà, per esempio

$$(a + ib)^2 + (c_x + ic_y) = (a^2 - b^2 + c_x) + i(2ab + c_y)$$

L'addizione e la moltiplicazione di numeri complessi godono delle usuali proprietà algebriche: associativa, commutativa, distributiva.

L'insieme di Julia su Commodore 128

```
5 GRAPHIC 1,1
10 X1=-1  : X2=1
20 Y1=-1  : Y2=1
30 DX=200 : DY=100
40 CX=-1
50 CY=0
100 FOR R=1 TO DX
110 : FOR C=1 TO DY
120 :   X=X1+R*(X2-X1)/DX
130 :   Y=Y1+C*(Y2-Y1)/DY
140 :   CT=0: M2=X*X+Y*Y
150 :   DO WHILE (M2<4) AND (CT<16)
160 :     RE=X*X-Y*Y+CX
170 :     IM=2*X*Y+CY
180 :     X=RE : Y=IM
190 :     M2=X*X+Y*Y
200 :     CT=CT+1
210 :   LOOP
220 :   IF (M2<4) THEN GOSUB 1000
230 : NEXT C
240 NEXT R
250 GET A$ : IF A$="" THEN GOTO 250
260 GRAPHIC 0
270 END
1000 DRAW 1,R,C
1010 RETURN
```

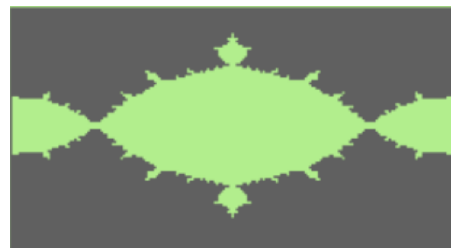



Figura 8. Insieme di Julia su C128

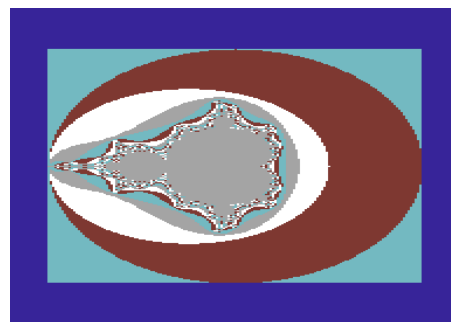
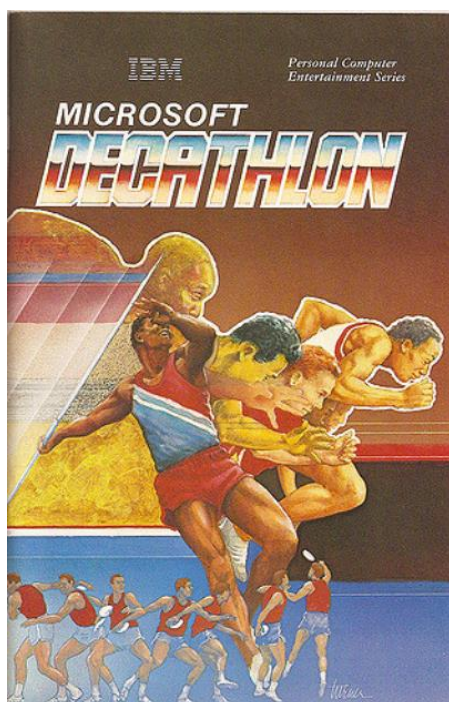



Figura 9. Frattale di Mandelbrot su C64

Insieme di Mandelbrot su Commodore 64 con Simon's Basic

```
1 REM ***********************
2 REM * COSTRUZIONE  *
2 REM * FRATTALE DI    *
3 REM * MANDELBROT   *
4 REM *                    *
5 REM * RIS. 160X200     *
6 REM * 4 COLORI          *
7 REM *********************
9 :
10 IX=-2:SX=2: REM PORZIONE DI PIANO COMPLESSO
20 IY=-2:SY=2: REM DA VISUALIZZARE
30 MC=32: REM NUMERO DI ITERAZIONI DEL CICLO DI CALCOLO
100 HIRES 1,0:MULTI 3,2,1: REM ABILITA AMBIENTE GRAFICO MULTICOLOR
110 :
120 LX=SX-IX: LY=SY-IY: REM AMPIEZZA INTERVALLO ASCISSE ED ORDINATE
130 XX=LX/160: REM INTERVALLO ASCISSE IN RAPPORTO ALLA RISOLUZIONE X
140 YY=LY/200: REM INTERVALLO ORDINATE IN RAPPORTO ALLA RISOLUZIONE Y
150 :
160 FOR I=0 TO 160 : REM CICLO PER CIASCUNA COORDINATA X
170 CR=IX+(I*XX) : REM CALCOLO PARTE REALE C
180 FOR J=0 TO 200 : REM CICLO PER CIASCUNA COORDINATA Y
190 CT=0: REM CONTATORE
200 ZI=0 : REM PARTE IMMAGINARIA DI Z
210 ZR=0 : REM PARTE REALE DI Z
220 CI=IY+J*YY : REM CALCOLO PARTE IMMAGINARIA DI C
230 :
239 REM APPLICA LA LEGGE DI MANDELBROT...
240 IF((ZR*ZR+ZI*ZI <=4) AND (CT<MC)) THEN GOTO 300
250 CL=CT:PLOT I,J,CL AND 2+1: REM TRACCIA PUNTO A COORDINATE I,J DI COLORE CL AND 3
260 NEXT J
270 NEXT I
280 GETA$:IFA$=""THEN 280
290 END
300 :
310 ZI=2*ZI*ZR+CI: REM CALCOLO NUOVA PARTE IMMAGINARIA DI Z
320 ZR=ZR*ZR-ZI*ZI+CR: REM CALCOLO NUOVA PARTE REALE DI Z
330 CT=CT+1: REM INCREMENTA CONTATORE
330 GOTO 240
```

Microsoft Decathlon

Microsoft Corporation - Anno 1982 - Piattaforma IBM PC

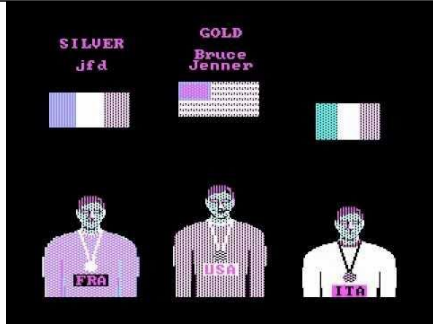
Marzo 1984, mio padre all'ingente cifra di 5.000.000(!) di Lire Italiane, compra il primo personal computer per tenere la contabilità della sua azienda ed io, facendo leva che era la settimana del mio compleanno, riuscii a convincere il venditore a regalarci un gioco. Visto che a quel tempo praticavo atletica leggera scelsi proprio Microsoft Decathlon.

Non vi dico la delusione quando scoprii che per farlo funzionare serviva un monitor CGA e non il monitor a fosfori verdi che aveva acquistato mio padre. Dovetti aspettare un anno prima di riuscire a giocarci, in quel periodo imparai il Basic e a programmarmi giochi con "grafica testuale", ma questa è un'altra storia...

La Microsoft creò il gioco nel 1980 per TRS-80 con il nome Olympic Decathlon e l'anno dopo lo traspose su Apple][e, quindi, nel 1982 su PC IBM cambiandogli nome in Microsoft Decathlon. Fu il precursore di Track & Field della Konami e di Activision Decathlon!

Il gioco, come dice il nome stesso, permetteva di gareggiare in tutte e 10 le specialità del decathlon olimpico: 100m, salto in lungo, getto del peso, salto in alto, 400m, 110m ostacoli, lancio del disco, salto con l'alta, lancio del giavellotto e, per finire, 1500m. Era possibile scegliere fino a 6 giocatori e per ciascuno di essi scegliere la nazionalità.

MICROSOFT DECATHLON

SILVER jfd
GOLD Bruce Jenner
FRA USA ITA

La premiazione

Anche se solo al terzo posto, siamo saliti sul podio!

Recensioni entusiastiche

All'epoca la maggior parte delle recensioni della carta stampata furono entusiastiche (Byte, PCMagazine, InfoWorld, solo per citarne alcune).

GIUDIZIO SUL GIOCO

GIOCABILITA'

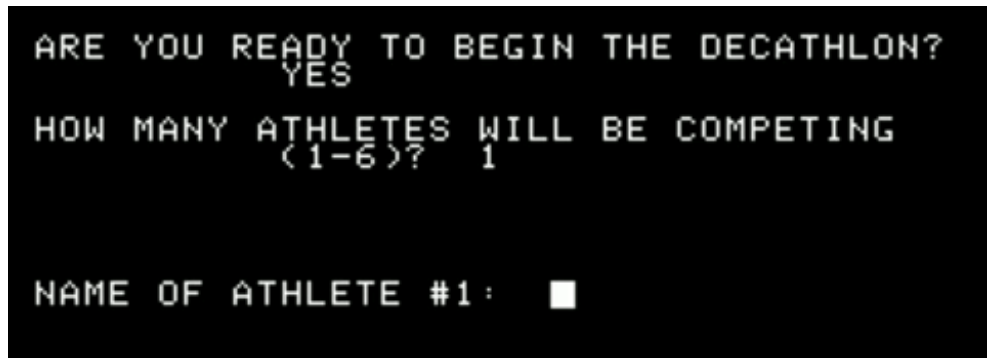
80%

Il gioco risulta divertente e avvincente malgrado le complicate combinazioni di tasti.

LONGEVITA'

50%

Malgrado fosse divertente, rimane negli annali piu' che altro perche' fu il primo gioco sportivo per PC.

ARE YOU READY TO BEGIN THE DECATHLON? YES
HOW MANY ATHLETES WILL BE COMPETING (1-6)? 1
NAME OF ATHLETE #1:

Si poteva anche scegliere se gareggiare in un solo sport, se gareggiare nel decathlon completo, oppure se ci si volesse solo "allenare".

Il controllo, ovviamente, avveniva tutto tramite tastiera; gli eventi di corsa erano quelli più semplici, bastava premere alternativamente i tasti "1" e "2" poi, a seconda della specialità, si dovevano aggiungere altri tasti, fino al più complicato, il salto con l'asta dove bisognava effettuare una combinazione di 5 tasti! Alla fine di ogni evento veniva assegnato un punteggio in base alla prestazione ottenuta (secondo il regolamento olimpico), la somma dei punteggi delle varie specialità forniva la classifica finale. E' interessante notare che nel gioco sono presenti i punteggi reali dei decatleti ottenuti durante le Olimpiadi dell'epoca, e il vincitore delle Olimpiadi del 1976, Bruce Jenner, è presente come personaggio… battere i punteggi reali degli atleti richiedeva ore di allenamento, combinando velocità, tempismo e precisione, proprio come nel vero decathlon.

La grafica, malgrado inferiore ad altri sistemi dell'epoca, per l'implementazione su PC IBM era tecnicamente avanzata perché era uno dei primi a sfruttare la modalità CGA composita e la Palette 0 (zero) e malgrado i soli 4 colori riusciva a creare combinazioni interessanti sul monitor RGB. In rete sono riuscito a trovare un jpeg che mostra le differenti grafiche dei 3 sistemi.

La musica, ovviamente, era praticamente inesistente; si limitava solo una musichetta gracchiata dall'altoparlante del PC durante la cerimonia d'apertura e la premiazione finale.

Malgrado la spartanità, il gioco risultava piacevolmente giocabile e assicurava ora di divertimento anche in compagnia. A me piaceva moltissimo e continuai, ogni tanto, a giocarci anche quando, all'inizio del 1986, riuscii a farmi regalare il Commodore 128.

Una curiosità: su Facebook esisteva fino a qualche tempo fa una pagina chiamata "I wrecked three keyboards playing Olympic Decathlon on the Apple II" (Ho rotto 3 tastiere giocando a Olympic Decathlon su Apple II) ho provato a cercarla nuovamente durante la scrittura di questo articolo, purtroppo senza successo.

di Gianluca Romani

ACTRAISER

Traduzione in ritardo...

L'idioma giapponese rendeva alcune quest di difficile risoluzione la traduzione arrivera' solo dopo qualche mese dopo.

Effetti speciali

Vanno evidenziati i numerosi strati di parallasse che donano profondità ad alcuni scenari e la trasperanza dell'acqua.

GIUDIZIO SUL GIOCO

GIOCABILITA'

95%

Da qualsiasi punto lo si guardi, non si riesce a trovargli un difetto e non esagero se dico che Actraiser è tuttora uno dei più grandi action mai concepiti.

LONGEVITA'

75%

Cosi' bello che vale la pena rigiocarlo di tanto in tanto.

Actraiser

Quintet - Anno 1990 - Piattaforma Super Famicom

Il lancio giapponese del Super Famicom sconvolse non poco i videogiocatori dei primi anni 90. Tra novembre e dicembre uscirono titoli dalla qualità media altissima, giochi che nelle classifiche dei generi videoludici andavano ad occupare tranquillamente il gradino più alto. La Nintendo ridefinì il genere dei platform con l'immenso Super Mario World e rivoluzionò le simulazioni arcade automobilistiche con l'innovativo (graficamente) e velocissimo F-Zero, senza contare lo stupore degli appassionati di simulazioni alla vista delle zoomate di Pilotwings. Oltre all'ottima (per l'epoca) conversione di Final Fight, i possessori di questa fortunata console poterono godere anche del gioco più originale dell'anno: Actraiser di Enix. Chi avrebbe mai immaginato che potesse risultare vincente un mix tra Rastan e Populous?

Actraiser è più che un valido gioco, qui ci troviamo di fronte a un vero e proprio capolavoro dai valori produttivi altissimi (nei titoli di coda ci sono una quarantina di persone!). Il gioco si compone di 2 fasi. La prima, simile a Populous è sicuramente la più debole, ma abbinata ai livelli action crea nel gioco un gameplay particolarmente originale. In questa prima fase strategica scendiamo tra i nostri fedeli guidandoli nella costruzione di città, aiutandoli con magie e risoluzioni di piccole quest ed uccidendo nel frattempo piccoli mostri sotto la forma di un angioletto. La cosa bella è che la trama non è lineare e spesso dobbiamo visitare alcuni luoghi già liberati, perchè in precedenza non avevamo l'oggetto che utile allo scopo. Come nei migliori RPG più crescono e si evolvono le città e più aumentano i valori del nostro guerriero. In questa fase l'interazione con i fedeli è continua, ci faranno offerte e non mancaranno richieste come ad esempio far ripartire le pale dei mulini a vento o ritrovare un familiare disperso.

In ogni regione sono presenti anche 2 fasi action, la prima affrontabile immediatamente, la seconda solo dopo che i capi della comunità avranno evocato il Master, per informarlo che è stata trovata la causa dei problemi che la affliggono (quando la città ha raggiunto un determinato grado di sviluppo). Queste fasi sono il vero fulcro del gioco, è qui che Actraiser dà il meglio di se.

La varietà degli scenari è straordinaria: foreste, laghi, deserti, pianure innevate e perfino l'interno di un vulcano. La sensazione di trovarsi in un vero e proprio mondo fantastico è qualcosa di meraviglioso. Ogni regione ha la propria fauna e i percorsi anche qui, spesso non sono lineari. I pittoreschi scenari sono una delizia per gli occhi e variano durante il medesimo stage. Forse gli sprite dei mostri non sono della stessa qualità, in alcuni casi sono piccoli e vantano poche animazioni, ma c'è talmente tanta carne al fuoco che questo difetto passa in secondo piano. Ottimi i boss di fine livello, non sono enormi ma offrono grande varietà e sopratutto hanno pattern di attacco differenti.

Dal mio punto di vista Actraiser stupisce più per le sue qualità artistiche che tecniche, anche se vanno evidenziati i numerosi strati di parallasse che donano profondità ad alcuni scenari e la trasperanza dell'acqua (una novità per l'epoca, solo il SF la possedeva). Senza contare inoltre la vastissima paletta di colori del Famicom, dubito che si arrivi ai 256 colori su schermo, ma non ricordo di aver mai visto una neve di tale corposità e anche gli oggetti d'oro sono resi straordinariamente bene. Sbalorditivo inoltre l'uso del mode 7 quando si sorvola la mappa. Aggiungete a tutto questo ben di dio la magistrale colonna sonora di Yuzo Koshiro (famoso grazie a Super Shinobi) ed avrete un gioco dall'atmosfera impareggiabile. Il particolare suono del chip sonoro del Super Famicom, pomposo e orchestrale come non mai rende le battaglie a dir poco epocali.

Terminare Actraiser all'epoca non fu facile, alcune fasi action erano veramente dure (per esempio le ultime 2) senza contare il combattimento finale che comprendeva lo scontro anche con i boss precedenti...

di Frenda Roberto

ACTRAISER

Presentazione notevole...

Una Lotus Esprit che riempie la videata con belle e dettagliate schermate informative.

Effetti speciali

Ogni circuito è pieno zeppo di curve di svariate angolazioni, dossi, cunette, salite, discese e serpentine con la linea dell'orizzonte che si estende fin dove nessun altro titolo automobilistico si era mai spinto prima.

GIUDIZIO SUL GIOCO

GIOCABILITA'

94%

Sicuramente il miglio arcade corsaiolo per Amiga del periodo, almeno fino all'avvento del suo seguito (il vero capolavoro di questa famosissima serie).

LONGEVITA'

90%

Il fascino di Lotus resiste all'azione del tempo!

Lotus Esprit Turbo Challenge

Magnetic Fields - Anno 1990 - Piattaforma Amiga

Su un vecchio numero di K la preview di Lotus che annunciava l'accordo tra la Gremlin e il celebre marchio automobilistico, parlava di due obiettivi principali dei programmatori: il gioco doveva essere migliore di qualsiasi altro gioco di guida in commercio e doveva offrire qualcosa di nuovo. Bisogna ammettere che una volta tanto le promesse furono mantenute: alla sua uscita Lotus è stato veramente il miglior gioco di guida disponibile su Amiga.

I possessori della macchina Commodore lo accolsero come il salvatore della patria; finalmente un gioco di guida su quattro ruote che rendesse giustizia alle potenzialità del loro 16 bit, specialmente dopo le amare delusioni dell'inverno precedente (Power Drift, Chase HQ e Turbo Outrun). Certo i detrattori potranno asserire che si tratta di un clone aggiornato del vecchio e intramontabile Pit Stop 2 del C64, ma splitscreen e sosta dei rifornimenti a parte, il titolo dei Magnetic Fields ha poco a che spartire con il titolo Epyx.

La prima cosa che colpisce di Lotus è l'ottima sequenza introduttiva: una Lotus Esprit che riempie la videata con belle e dettagliate schermate informative. Grandiosa anche l'idea dell'autoradio che, come in una vera macchina, ci permette di scegliere le tracce musicali che più ci piacciono (buone ma non ai livelli dei capitoli successivi).

La varietà delle 32 piste (divise in 3 livelli di difficoltà) è ottima, quasi ogni nazione in cui si corre possiede cartelli pubblicitari diversi ai lati della strada così come la propria vegetazione. Inoltre in molti tracciati si incontrano ostacoli come pozze d'acqua, macchie d'olio, transenne, segnali stradali ecc. Tutti questi oggetti si muovono con un ottima fluidità, grazie anche ai numerosi frames di animazione (sembra quasi di trovarsi davanti ad un coin-op Sega!). Per non parlare di come sono disegnate le piste, ogni circuito è pieno zeppo di curve di svariate angolazioni, dossi, cunette, salite, discese e serpentine con la linea dell'orizzonte che si estende fin dove nessun altro titolo automobilistico si era mai spinto prima (solo F-Zero del Super Famicom riuscì quello stesso anno a fare di meglio).

Tutto questa magnificenza grafica sarebbe sprecata senza un'adeguata velocità e Lotus la possiede. Anzi, direi di più, è velocissimo, una velocità mai vista prima su Amiga (più veloce anche di Super Hang-on). Inoltre non rallenta o scatta mai, nemmeno quando la pista è strapiena di macchine (alla partenza sono la mostruosa cifra di 20!!!).

Se Lotus è bello da vedere, da giocare è ancora meglio. Il divertimento non è dato solo dalla velocità e dalla conformazione delle piste, ma anche dai numerosi sorpassi da effettuare, magari mentre si evitano i fastidiosi ostacoli. Tutto questo sempre tenendo d'occhio la benzina e la posizione in gara della nostra macchina. Nei circuiti avanzati mi sono ritrovato a non classificarmi tra i primi 10 (pena l'eliminazione) proprio perchè le macchine mi superavano dopo i rifornimenti.

Nel modo a 2 giocatori si sfiorano vette di giocabilità altissime, roba che solo un classico intramontabile come Pit Stop 2 aveva saputo offrirci sugli home computer. Certo la macchina durante gli scontri rallenta e basta, non ci sono incidenti o ribaltamenti alla Outrun e questo potrebbe far sembrare il gameplay fin troppo semplice e arcade. Ma è un impressione sbagliata in quanto l'automobile è molto sensibile alla morfologia dell'asfalto; il motore ci mette un sacco di tempo a prendere i giri quando si tampona qualcosa durante le numerose salite della pista. Diciamo che molti circuiti permettono un bel po' di errori, ma in alcuni di essi, se si vuole arrivare primi, bisogna guidare alla perfezione! Per quanto riguarda la difficoltà, finire i livelli easy e normal è abbastanza facile, quello hard però è molto impegnativo (specialmente l'ultima pista Antarctica, dove è gia' un impresa qualificarsi tra i primi 10) non fosse altro per la sua durata che si aggira sull'oretta o poco più… e non si può salvare…

di Frenda Roberto

GLORIOUS FORCE – THE FORCE OF DARKNESS

Glorious Force

La presentazione era abbastanza scarna…

Effetti speciali

Il gioco e' scritto interamente in BASIC.

GIUDIZIO SUL GIOCO

GIOCABILITA'

80%

Stiamo parlando si un gioco scritto in basic con le dovute limitazioni del linguaggio ma abbastanza intrigante. Risulta lineare come storia se lo vediamo con la esperienza attuale sui mondi RPG ma non noiosa.

LONGEVITA'

75%

Solito tallone di achille dei giochi rpg.. una volta finiti difficile rigiocarli, ma questo game può piacere rigiocarlo in quanto come lunghezza di game play non è eccessiva.

Glorious Force - The force of Darkness

Aquarius - Anno 1987 - Piattaforma MSX

Nel mondo dei videogiochi e' da sempre esistita la categoria dei role play game o adventure… il sottoscritto neofita dell'MSX ma conoscitore dei sistemi TI99/4A ed Amiga, che di queste categorie di giochi hanno visto delle vere e proprie pietre miliari, vedi Tunnels of Doom per TI99 e Dungeon Master per Amiga, ha cercato per il sistema MSX dei giochi RPG per capire se furono creati con una concezione diversa rispetto ai due computer sopracitati. Stiamo sempre parlando di una macchina MSX1 che è un 8 bit di poco superiore al TI99 ma inferiore all'Amiga, macchiana concepita con idee rivoluzionarie.

Nel cercare questo tipo di giochi ho scavato nell'immenso archivio dei sistemi MSX scoprendo con disappunto che di giochi RPG ne hanno fatti a decine ma al 95% tutti in Giappone e solo in giapponese! Questo mi ha creato una grossa limitazione nella scelta dei giochi ma sono riuscito a scoprire due giochi RPG che sono stati creati con idee innovative rispetto a quelli creati in Europa o negli USA.

Per giudicarli li ho giocati a fondo e portati a termine in modo da scoprirne tutte le loro peculiarita'.

I titoli che ho scelto creati in Giappone ma con una traduzione in inglese, sono Hydlide, recensito nel numero 2, e Glorious Force creato nel 1987 da un programmatore sotto lo pseudonimo di Aquarius.

Con Glorious Force abbiamo, dal mio punto di vista, una vera e propria perla di programmazione relativamente al game play ed alla macchina. Nulla di nuovo od eccezionale, ma considerando il fatto che il gioco fu scritto in BASIC da un singolo programmatore, il quale conoscendo le limitazioni del computer ma facendo tesoro dei nuovi modi di concepire il game play ha creato questo RPG in formato simil 3D isometrico dove l'unico sprite in movimento è l'eroe. La mappa scorre solo orizzontalmente, idem per i vari dungeon e l'incontro/scontro con i vari mostri viene solamente descritto con un sistema a turni ma dando al giocatore la possibilità di interagire nel combattimento, cosa che di oggi e di fatto uno standard. Il gioco è di media difficoltà, abbastanza intrigante e si deve fare un poco di strategia gestionale del denaro per capire come attrezzarsi per la bisogna. Peccato che come al solito questo gioco creato nel 1987 non è mai arrivato qui in Europa, avrebbe meritato. Come game play Glorious Force è molto vicino a Dragon Quest ma non per questo il gioco ne esce sminuito anzi esattamente il contrario si può dire che grazie a questo game play Glorious Force abbia acquistato uno spessore e profondità.

Con oggi abbiamo chiuso dell'argomento RPG MSX1 ma la discussione è stata toccata solamente in minima parte, se riuscirò ad avere altri RPG per Msx che siano giocabili spero di fare altre recensioni su essi.

di Ermanno Betori

DRACULA'S CASTLE

Musiche di pregio

Godetevi l'ispiratissimo motivetto dai toni tipici dei migliori horror classici.

Effetti speciali

Il titolo non è un prodotto originale, bensì il remake di un gioco del 1985 a cui però sono stati aggiunti una schermata finale, candele in movimento, un punteggio di Hi-Score e tanti altri particolari che vi sfido a scoprire.

GIUDIZIO SUL GIOCO

GIOCABILITA'

80%

Vi serviranno alcune partite per entrare nel ritmo del gameplay e comprendere quanto il tempismo del sali-scendi dalle scale si riveli fondamentale per sopravvivere, ma una volta presa la mano scoprirete un gioco davvero spassoso e intuitivo.

LONGEVITA'

65%

Quattro livelli sono effettivamente molto pochi ma l'inseguimento del miglior punteggio e la sfida a terminare il tutto senza mai morire ve lo renderanno ben rigiocabile.

Dracula's Castle

MP Software - Anno 2017 - Piattaforma C64

Innanzi tutto una premessa: non ho mai avuto un Commodore 64. Certo, ci ho giocato a lungo a casa di amici, eppure non ne ho mai posseduto uno. Vi sembra saggio dare ad un tizio così poco raccomandabile il compito di recensire un nuovo gioco per il più importante home computer ad 8-bit della storia? Domande retoriche a parte la verginità ha anche i suoi pregi ed infatti mi sono armato di Joystick Albatross (opportunamente configurato su raspberry pi 3) e motivatissimo eccomi ad attraversare le varie stanze del castello di Dracula, con il chiaro obiettivo di uscirne vivo.

Il gioco è un platform a schermata fissa che ci pone al controllo di un malcapitato omino poco propenso al combattimento e per tanto costretto a scappare di stage in stage.

Muovendo il nostro joystick a destra o sinistra ci muoveremo nella tal direzione, facciamolo mentre premiamo il tasto Fire e ci sposteremo saltando. Premendo in basso e Fire potremo accucciarci per evitare i pipistrelli o il Boss del terzo livello (poi non dite che non vi aiuto) mentre la semplice pressione del tasto in prossimità di un oggetto ci permetterà di raccoglierlo. In ultimo, se ci porremo nei pressi di una scala potremo salirla o discenderla con la levetta in su o in giù.

Gli oggetti che potremo raccogliere sono di due tipi: croci o chiavi. Le prime ci permetteranno di respingere temporaneamente i nemici con cui verremo a contatto (consumandosi), mentre la chiave ci servirà in ogni livello per poter aprire la porta che conduce al successivo stage. I nemici presenti, fatta eccezione per il suddetto boss, sono principalmente di due tipi: pipistrelli giganti e Dracula. I primi potranno essere elusi genuflettendoci con rapidità mentre il secondo andrà tenuto a buona distanza se non ci si è riforniti di tante belle croci in puro stile B-movie anni 50 (la Hammer ringrazia). State attenti infine ai teschi disseminati lungo il percorso perché se non saltati ci inciamperemo e saremo impossibilitati a rialzarci per alcuni pericolosissimi secondi.

Immediato, divertente e completamente gratuito, mi sento di consigliarlo sia ai vecchi appassionati che ai neofiti che volessero approcciarsi al biscottone partendo dall'emulazione (ogni riferimento a me stesso è puramente casuale). Come avrete modo di leggere anche nell'intervista al suo creatore, Dracula's Castle è una produzione squisitamente italiana che vede la firma di **Marco Pistorio**, un crociato del retrogaming che condivide con tutto il mondo la sua creazione http://csdb.dk/release/?id=160683 e scrive sulla nostra rivista. Conflitto di interessi? Forse, ma magari è proprio per questo che mi sono assunto io la responsabilità di recensirlo, perché più neutrale di uno che non ne sa nulla non so chi speravate di trovare.

di Starfox Mulder

Intervista a Marco Pistorio

PROGRAMMATORE DI DRACULA'S CASTLE PER COMMODORE 64

Ciao Marco, domanda di rito: cosa ti ha motivato a programmare un titolo per commodore 64 nel 2017?

Ciao Starfox, ottima domanda (ride). Sono stato fermo per più di vent'anni relativamente allo sviluppo su C64 poi, su Facebook, trovai parecchia gente appassionata, diversi gruppi molto attivi che in un certo senso mi hanno "contagiato" questa voglia, questo desiderio di rimettermi in gioco. Dall'ottobre dell'anno scorso ho cercato di riprendere, esattamente da dove avevo lasciato tanti anni fa, prima con semplici demo e poi con qualche gioco.

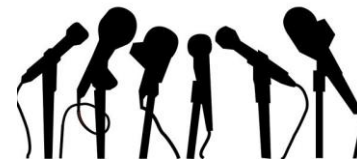
Il gioco Dracula's Castle è una sorta di remake in assembly del classico programmato in basic da Emerich Fernandes nel 1985. Puoi dirci in cosa differisce la versione da te programmata?

Relativamente al gioco Dracula's Castle posso dirti che, rispetto al gioco in basic, le differenze sono molteplici. Innazitutto, questo è scritto in assembly. Non convertito, ma riscritto, istruzione per istruzione. Grazie alla velocità dell'assembly il gioco è più "giocabile", il controllo è decisamente migliore. Ho introdotto alcune modifiche nella logica del terzo livello poichè nella versione originale risultava non superabile, una scena finale particolare che manca del tutto nella versione originale ed un bonus legato al tempo di gioco. E' stato previsto anche l'hi-score, con relativa gestione, insieme ad una base musicale decisamente più consona ai nuovi ritmi di gioco più qualche altro dettaglio qua e là come ad esempio le luci con un bagliore più "dinamico" ed un sistema di frecce che aiutano il giocatore a comprendere quando può salire le scale, scenderle e prendere oggetti.

Per la creazione del gioco c'è stata una bella collaborazione con altri artisti come ad esempio BAT che si è occupato delle musiche. Come avete coordinato il lavoro tra di voi?

La base musicale l'ho scelta personalmente all'interno delle basi offerte dal sito hvsc in formato .sid. Incorporarle è una cosa decisamente semplice in assembly, ed è ancora più facile adoperando KickAssembler oltretutto. Non è stato quindi necessario coordinarmi con l'autore del file .sid ma è stato giustamente menzionato nei crediti all'interno del gioco. Per quanto riguarda invece il supporto grafico, luci candele, sprites frecce e pic di presentazione in particolar modo, mi sono avvalso della collaborazione di Federico Sesler, anche lui menzionato ovviamente nei crediti all'interno del gioco. Qui in effetti è stato necessario coordinarci, sempre grazie a Facebook.

Ti confesso che mi sono molto divertito a giocare e finire il gioco, ho subito avuto voglia di riprovarci migliorando la mia prestazione ma effettivamente quattro livelli sono pochi di fronte ad un gameplay così accattivante. Hai mai pensato di ampliare il gioco con una versione "estesa"?

L'idea di una versione estesa in effetti l'ho avuta, suggerita anche dall'amico Federico Sesler. In genere quando intervengo su dei giochi BASIC cerco di lasciarne inalterata la loro logica e la loro genuinità, sottolineando al contempo la loro potenzialità, spesso sottovalutate a causa dei limiti intrinseci legati al linguaggio BASIC. Spero di essere riuscito a fare tutto ciò anche con Dracula's Castle. Nella versione extended, se la porterò avanti, lo scopo potrebbe essere quello di rivedere in maniera più incisiva grafica, logica, livelli, scene, eccetera. Vedremo con il nuovo anno.

Ti ringrazio della gentilezza e azzardo un ultima domanda: ci sono già nuovi giochi in cantiere?

Sì. Ho ripreso proprio in questi giorni un giochino, un clone del famoso "Tetris" che scrissi in assembly oltre vent'anni fa, esattamente nel 1994. Ho trovato nei miei dischetti i modelli BASIC dai quali sono partito ed il codice assembly. La mia intenzione è di pubblicarlo presto e di presentarlo poi anche ai lettori di "RetroMagazine", con tutta la necessaria documentazione, codice e commenti del caso.

Ancora grazie Marco, è stato un piacere.

Piacere mio, Starfox. Spero che le mia produttività e la mia voglia di fare e mettermi in gioco continui ancora a lungo così, e magari cresca, chissà. Ciao e alla prossima!

di Starfox Mulder

Contatti
Se avete qualche domanda particolare da rivolgere a Marco riguardo il gioco Dracula's Castle o sulla programmazione Assembly in generale, potete scrivere a Marco tramite la mailbox redazionale:
RetroMagazine.redazione@gmail.com

Sui prossimi numeri...

Ed anche questo numero 3 di RetroMagazine è giunto alla fine.

Ringraziamo tutti i nostri lettori, uno per uno, per averci seguito in questi ultimi mesi del 2017. Ci auguriamo che sarà così anche per tutto il nuovo anno 2018 appena iniziato. Inoltre ci auguriamo che sarete in tanti a seguirci, che sarete sempre di più e sempre più soddisfatti del nostro lavoro e del nostro impegno. Vi promettiamo che ce la metteremo tutta, con le nostre recensioni, le nostre rubriche, i nostri articoli, le nostre interviste, e tutto ciò che ci verrà in mente strada facendo!

A proposito... abbiamo bisogno di Voi! Scriveteci! Inondate la mail redazionale con le Vostre lettere! I Vostri consigli, i Vostri suggerimenti, il Vostro giudizio su quello che trovate di bello e di meno bello in "RetroMagazine" sarà un prezioso strumento per aiutarci a fare sempre meglio. Chiedeteci degli approfondimenti, poneteci le Vostre domande. Saremo lieti di pubblicarle in una nuova rubrica, "**L'angolo della Posta**", che allestiremo già a partire dai prossimi numeri.

Cercheremo di rispondere a tutti ovviamente nei limiti del possibile.

Adesso le anticipazioni su alcuni degli argomenti dei prossimi numeri:

Francesco ci parlerà finalmente del GEOS o del CP/M sul C128 (devo ancora decidere... NdFF).
Marco invece illustrerà come generare toni DTMF in BASIC con il Commodore 64. **Leonardo Vettori** ci racconterà del recente evento "Firenze Vintage Bit 2017". **Giorgio**, dopo una doverosa introduzione dell'hardware dell'Atari 2600, ci insegnera' a programmare in un linguaggio poco conosciuto, il Batari... Curiosi? Non perdetevi i prossimi numeri di RetroMagazine. Qualcuno ha detto Raspberry PI? Cosa c'entra con il RetroComputing? Chiediamolo a **Dante**...

E non è mica finita qui! Ma non possiamo anticiparVi tutto :) I più sinceri auguri a tutti Voi ed alle Vostre famiglie di un felice nuovo anno 2018 dalla redazione di RetroMagazine e... a presto!

Come di consueto ormai da un paio di numeri, se volete scriverci per suggerimenti, commenti o richieste per i prossimi articoli, potete farlo tramite la casella postale: retromagazine.redazione@gmail.com

Da ora in poi questo sara' anche l'indirizzo per le lettere con le vostre domande per l'angolo della posta. Approfittatene!

Disclaimer

RetroMagazine
Anno 2 - Numero 3

Direttore Responsabile
Fiorentini Francesco

Immagine di copertina
Flavio Soldani

Gennaio 2018